Using the Terminal

Version 2020.3
2020-12-10
# Table of Contents

About This Book ................................................................................................................................. 1

1 Introduction to the Terminal ............................................................................................................. 3
   1.1 User Account That Owns the Terminal Process ................................................................. 3
   1.2 Starting the Terminal .............................................................................................................. 3
   1.3 Background ............................................................................................................................... 4
   1.4 General Use ............................................................................................................................... 4
   1.5 The ZWELCOME Routine ..................................................................................................... 5
   1.6 The Startup Namespace ......................................................................................................... 5
       1.6.1 Changing Namespaces .................................................................................................. 5
   1.7 The Terminal Prompt .............................................................................................................. 5
       1.7.1 The Transaction Level ................................................................................................. 6
       1.7.2 The Program Stack Level ............................................................................................ 6
       1.7.3 The SQL and MDX Shells ........................................................................................... 6
       1.7.4 Operating-System Shells ............................................................................................. 6
       1.7.5 Customizing the Terminal Prompt ............................................................................. 6
   1.8 Interrupting Execution in the Terminal .................................................................................. 7
   1.9 Exiting the Terminal .............................................................................................................. 7

2 Using the Terminal Interactively .................................................................................................. 9
   2.1 Entering Commands ............................................................................................................... 9
   2.2 Scrolling ................................................................................................................................. 9
   2.3 Pausing and Resuming the Terminal Scrolling .................................................................... 10
   2.4 Repeating Previous Commands ......................................................................................... 10
       2.4.1 Line Recall ................................................................................................................... 11
       2.4.2 Reverse Incremental Search ....................................................................................... 11
   2.5 Copying and Pasting Text ..................................................................................................... 12
       2.5.1 Keyboard Shortcuts .................................................................................................... 12
       2.5.2 Notes about Copying and Pasting ............................................................................. 13
   2.6 Printing .................................................................................................................................. 13
   2.7 Clearing the Screen .............................................................................................................. 13
   2.8 Logging the Terminal Session .............................................................................................. 13

3 Controlling the Appearance and Behavior of the Terminal ........................................................... 15
   3.1 Specifying the Font ............................................................................................................... 15
   3.2 Specifying the Colors ......................................................................................................... 15
   3.3 Specifying the Window Size ............................................................................................... 16
   3.4 Defining Custom Key Combinations .................................................................................... 16
   3.5 Specifying Other User Settings ......................................................................................... 16
   3.6 Specifying the Network Encoding ....................................................................................... 17
       3.6.1 UTF8 Encoding ........................................................................................................... 18
       3.6.2 Windows Encoding .................................................................................................... 18
       3.6.3 ISO Encoding ............................................................................................................. 18
       3.6.4 EUC Encoding .......................................................................................................... 19
   3.7 Specifying the Physical Character Setting of the Display .................................................. 19

4 Using Terminal Scripts .................................................................................................................. 21
   4.1 Starting Script Execution ..................................................................................................... 21
   4.2 Pausing Script Execution .................................................................................................... 21
4.3 Stopping Script Execution ................................................................. 21
4.4 Contents of a Script File ...................................................................... 22
4.5 Script Command Summary ............................................................... 22
4.6 Script Command Arguments .............................................................. 23
4.7 A Sample Script ................................................................................ 24

5 Script Command Reference .................................................................. 27
5.1 break .................................................................................................. 27
5.2 call script .......................................................................................... 27
5.3 case match ......................................................................................... 27
5.4 closelog ............................................................................................. 28
5.5 connect .............................................................................................. 28
5.6 debug .................................................................................................. 28
5.7 disconnect ......................................................................................... 28
5.8 display ............................................................................................... 28
5.9 echo ..................................................................................................... 28
5.10 execute ............................................................................................. 29
5.11 exit .................................................................................................... 29
5.12 goto .................................................................................................. 29
5.13 if empty ............................................................................................ 30
5.14 key_starttime .................................................................................. 30
5.15 key_stoptime ................................................................................... 30
5.16 key_timer ......................................................................................... 30
5.17 logfile .............................................................................................. 31
5.18 multiwait for ................................................................................... 31
5.19 notify ............................................................................................... 31
5.20 on error ............................................................................................ 32
5.21 pause ............................................................................................... 32
5.22 return ............................................................................................... 32
5.23 send ................................................................................................. 32
5.24 subroutine ....................................................................................... 33
5.25 terminate ......................................................................................... 33
5.26 test ................................................................................................... 33
5.27 timer ............................................................................................... 34
5.28 title ................................................................................................. 34
5.29 wait for ............................................................................................ 34

6 Connecting to Remote Hosts ............................................................. 37
6.1 Example ............................................................................................ 37

7 Using the Terminal in Batch Mode ..................................................... 39
7.1 The Batch Command Line ............................................................... 39
7.2 Control Arguments ........................................................................... 39
7.2.1 /console=<ConnectString> ......................................................... 40
7.2.2 /server=<ServerName> ............................................................... 40
7.2.3 /size=RowsxCols ........................................................................ 41
7.2.4 /pos=(X,Y) ................................................................................ 41
7.2.5 /ppos=(Xpct,Ypct) ...................................................................... 41
7.2.6 /UnbufferedLogging ................................................................... 41
7.3 Examples ........................................................................................ 41
7.3.1 Running a Script in Batch Mode ................................................. 42
7.3.2 Running a Script Interactively .................................................... 42
8 Advanced Topics ............................................................................................................................... 43
  8.1 Disabling the Close Button of the Terminal ................................................................. 43
  8.2 Mapping of the Extended Keyboard ............................................................................. 43
  8.3 Special Modes .............................................................................................................. 44
    8.3.1 Key Timing Mode ...................................................................................................... 44
    8.3.2 Learning Mode ......................................................................................................... 44
  8.4 Using the Terminal with DDE .................................................................................. 44
    8.4.1 DDE Layout Connections ....................................................................................... 45
    8.4.2 DDE Screen Connections ....................................................................................... 45
    8.4.3 DDE Message Connections ..................................................................................... 45
List of Tables

Table 2–1: Using the Keyboard to Scroll the Terminal Display .............................................................. 9
Table 2–2: Shortcuts for Copy and Paste ............................................................................................... 12
Table 3–1: User Settings ...................................................................................................................... 17
Table 3–2: ISO Encoding ..................................................................................................................... 18
Table 3–3: EUC Encoding ..................................................................................................................... 19
Table 4–1: Terminal Script Commands ................................................................................................. 22
Table 4–2: Special Characters ............................................................................................................... 23
Table 7–1: Invoking the Terminal in Batch Mode ................................................................................. 39
Table 8–1: Key Mappings ...................................................................................................................... 43
Table 8–2: Keyboard Mappings ............................................................................................................. 44
Table 8–3: DDE Layout Connections ................................................................................................. 45
Table 8–4: DDE Screen Connections ................................................................................................. 45
Table 8–5: DDE Message Connections ................................................................................................. 46
About This Book

This book describes how to use the Terminal, the command-line interface for InterSystems IRIS® data platform. It contains the following chapters:

- Introduction to the Terminal
- Using the Terminal Interactively
- Controlling the Appearance and Behavior of the Terminal
- Using Terminal Scripts
- Script Command Reference
- Connecting to Remote Hosts
- Using the Terminal in Batch Mode
- Advanced Topics

For a detailed outline, see the table of contents.
1

Introduction to the Terminal

The Terminal is a simple command-line interface for entering ObjectScript commands and displaying current values. It is useful during learning, development, and debugging.

Note: If the Terminal displays a dialog box with the message Spy Mode On, this means that you have accidentally pressed Alt+Shift+S. To exit this mode, press Alt+Shift+S again. This mode is not for general use and is not documented.

Also, if the Terminal appears to be unresponsive, you may have pressed Ctrl+S, which pauses the automatic scrolling. If so, press Ctrl+Q to resume.

1.1 User Account That Owns the Terminal Process

The process is owned by the user that is logged in to Windows and is running the Terminal program (iristerm.exe).

Also, all environment variables and shared drive letter designations are those defined by the user that is running the Terminal.

1.2 Starting the Terminal

You can use the Terminal interactively or in batch mode.

To use the Terminal interactively, do one of the following:

• To work with the Terminal using a local database, select the InterSystems Launcher and then select Terminal.

• To work with the Terminal using a database on a remote server, select the InterSystems Launcher and then select Remote System Access > Terminal. Then select a server name.

Or select the InterSystems Launcher, select Remote System Access > Telnet. Then log on with your username and password.

For more information and additional options, see the chapter “Connecting to Remote Servers” in the System Administration Guide.

In either case, you then see the Terminal window. The default prompt displayed indicates the namespace in which you are currently working. For example:

USER>
In batch mode, you invoke the Terminal from the operating system command line, passing to it the name of a script file to run. This mode is not available for all operating systems.

### 1.3 Background

The Terminal was designed to work with InterSystems IRIS® applications. It uses two methods to communicate with InterSystems IRIS: local and network. The title bar indicates the communication mode currently in use.

- **Local communication** is used when the Terminal communicates with the InterSystems IRIS server with which it was installed. In this case, the title bar displays `InterSystems IRIS TRM: pid(instancename)` where:
  - *pid* is the process ID of the InterSystems IRIS process with which the Terminal is communicating.
  - *instancename* is the InterSystems IRIS instance in which the process is running.

- **Network communication** uses the TELNET protocol over TCP/IP to communicate with either a Windows InterSystems IRIS server or with a UNIX® host. In this case, the title bar displays `(server NT — InterSystems IRIS Telnet)` where *server* is the host name of the remote server.

The communications stack for InterSystems IRIS is Winsock, and errors reported from this communications mode are the names of the Winsock error codes. For example, `WSAECONNREFUSED` means the connection was refused.

### 1.4 General Use

In the Terminal, you can enter ObjectScript commands of all kinds. The Terminal does not support multi-line mode, so the entire command must be typed on a single line. For example:

```objectscript
D "myroutine"

set dir = "c:\test"

set obj = ##class(Test.MyClass).%New()
write obj.Prop1
```

**Note:** The Terminal implicitly issues the `Use 0` command after each line you enter. This means that if you issue a `Use` command to direct output to some other device, that command is essentially ignored.

Also, large input buffers may defer the action of keys that attempt to stop the input flow such as `Ctrl-C` or `Ctrl-S`. This is also dependent on processor and connection speed. A special effort was made to respond to keystrokes before host input.

You can also run **Terminal scripts**, which are files with the extension `.scr` existing in your file system. The Terminal provides a small set of commands you can use in these scripts, including a command that sends an ObjectScript command to the Terminal, as if you had typed it manually.
1.5 The ZWELCOME Routine

When the Terminal begins execution, the code checks for the existence of a routine called ZWELCOME in the %SYS namespace. If such a routine is found, it is invoked immediately prior to the terminal login sequence, if any. The name of the routine implies its intended use, as a custom identification and welcome message to users.

**Note:** The installation of ZWELCOME into the %SYS namespace requires an individual with administrator privileges and write access to the IRISSYS database.

**CAUTION:** The ZWELCOME routine executes in the %SYS namespace with an empty $USERNAME and with $ROLES set to %ALL. Take care to ensure that the failure modes of ZWELCOME are benign. Also, this routine should not modify the $ROLES variable.

Here is a simple example:

```
ZWELCOME() PUBLIC ;
; Example
Write !
Set ME = ##class(%SYS.ProcessQuery).%OpenId($JOB)
Write "Now: ", $ZDATETIME($HOROLOG, 3, 1), !
Write "Pid/JobNo: ", ME.Pid, "/", ME.JobNumber, !
Write "Priority: ", ME.Priority, !
Quit
```

1.6 The Startup Namespace

When you start the Terminal, it opens in a particular namespace. This option is controlled by the Startup Namespace option of the user definition. See the chapter “Users” in the Security Administration Guide.

The default prompt displays the current namespace, such as:

```
USER>
```

1.6.1 Changing Namespaces

To change to a new namespace, use the $namespace variable:

```
USER>set $namespace="SAMPLES"
SAMPLES>
```

See the $namespace reference page in the ObjectScript Reference for more information.

1.7 The Terminal Prompt

The default prompt in the Terminal indicates the namespace in which you are currently working. The prompt may display additional information, to indicate the transaction level or the program stack level. If you are using an SQL shell, the MDX shell, or an operating-system shell, the prompt changes. You can also customize the default prompt.
1.7.1 The Transaction Level

If you are within a transaction, the prompt includes a prefix that indicates the transaction level. The prefix is of the form TLn:, where n is the transaction level. For example, if you are in the User namespace and you enter the ObjectScript command TSTART, the prompt changes as follows:

USER>tstart
TL1:USER>

If you exit the Terminal, that rolls back the transaction.

1.7.2 The Program Stack Level

If an error occurs, the prompt includes a suffix that indicates the program stack level. For example:

USER 5d3>

Enter the Quit command to exit the debug prompt. Or debug the error; see the chapter “Command-line Routine Debugging” in the book Using ObjectScript.

1.7.3 The SQL and MDX Shells

In the Terminal, you can use shells that enable you to query data in various ways. When you use any of these shells, the prompt changes as shown in the table below:

<table>
<thead>
<tr>
<th>Shell</th>
<th>Change in the Prompt</th>
<th>Also See</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL</td>
<td>[SQL] prefix is added</td>
<td>“Using the SQL Shell Interface” in Using InterSystems SQL</td>
</tr>
<tr>
<td>TSQL</td>
<td>[SQL] prefix is added</td>
<td>“Overview” in Transact-SQL (TSQL) Migration Guide</td>
</tr>
<tr>
<td>MDX</td>
<td>Prompt becomes &gt;&gt;</td>
<td>“MDX Shell” in Introduction to InterSystems IRIS Business Intelligence</td>
</tr>
</tbody>
</table>

1.7.4 Operating-System Shells

In the Terminal, you can also open your default operating-system shell. To do so, type ! or $ and press Enter. The Terminal then opens the default operating-system shell, and the prompt shows the working directory. For example:

USER>!
c:\intersystems\iris\mgr\user\>

**Note:** On Macintosh, you cannot open the C-shell this way; you receive a permission denied error. You can, however, use other shells (Bash, Bourne, or Korn).

To exit the shell, use the quit or exit command as appropriate for the shell.

1.7.5 Customizing the Terminal Prompt

To customize the default Terminal prompt, use the Management Portal to set the configuration option TerminalPrompt, which controls the Terminal prompt. This option is available in the Management Portal. To find it, select System Administration > Configuration > Additional Settings > Startup. For example, the Terminal prompt can include the configuration name of your system.
1.8 Interrupting Execution in the Terminal

To interrupt the Terminal and stop any foreground execution, use one of the following key combinations:

- **Ctrl+C** — Use this if the *Windows edit accelerators* option is not enabled.
- **Ctrl+Shift+C** — Use this if the *Windows edit accelerators* option is enabled.

For information on *Windows edit accelerators* option, see the section “User Settings,” in the chapter “Controlling the Appearance and Behavior of the Terminal.”

1.9 Exiting the Terminal

To exit the Terminal, do any of the following:

- Select **File > Exit**
- Press **Alt+F4**
- Enter **HALT** or **H** (not case-sensitive)

These options cause this copy of the Terminal to exit, closing any open files and stopping any foreground execution. If this Terminal was connected to a server at startup, it exits on its own when the communications channel is closed.

If you accessed this Terminal via **InterSystems Telnet** in the InterSystems Launcher, then it does not exit automatically when the communications channel is closed; instead it remains active so you that can connect again via the **Connect** menu.
Using the Terminal Interactively

This chapter describes the basics of how you work with the Terminal in interactive mode.

2.1 Entering Commands

To enter a command in the Terminal, simply type an ObjectScript command and its arguments. Or, if you are using a shell, type a command applicable to that shell.

Note that for ObjectScript, you do not need to enter a space before the command (in contrast to when you define routines and methods).

Also note that if you start a line in the Terminal by pressing Tab, the rest of that line is entered into the routine buffer and is not executed. This behavior was defined long ago and was a useful way of creating routines. (Also see the ZINSERT command, which serves the same purpose.) Because customers no longer define routines this way, this documentation does not discuss this feature further.

2.2 Scrolling

Whenever active text arrives, the Terminal scrolls the window to the newly arrived text. Use the scroll bar on the right to scroll up or down.

You can also use the keyboard to scroll within the Terminal as follows:

<table>
<thead>
<tr>
<th>Key Combination</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+Home</td>
<td>Scroll to the top of the buffer.</td>
</tr>
<tr>
<td>Ctrl+End</td>
<td>Scroll down to the cursor.</td>
</tr>
<tr>
<td>Ctrl+Page Up</td>
<td>Scroll up by one page.</td>
</tr>
<tr>
<td>Ctrl+Page Dn</td>
<td>Scroll down by one page.</td>
</tr>
<tr>
<td>Ctrl+Line Up</td>
<td>Scroll up by one line.</td>
</tr>
<tr>
<td>Ctrl+Line Dn</td>
<td>Scroll down by one line.</td>
</tr>
</tbody>
</table>
2.3 Pausing and Resuming the Terminal Scrolling

To pause the scrolling in Terminal, press **Ctrl+S**. While scrolling is paused, the Terminal accepts commands and processes them, but it does not write the commands or any output to the screen (and thus appears to be unresponsive).

To resume, press **Ctrl+Q**.

2.4 Repeating Previous Commands

To repeat a previous command, press the up arrow key repeatedly until the terminal displays the desired command. To enter the command, press **Enter**.

2.4.1 Line Recall

The terminal also supports a more extensive line recall facility, which uses a list of commands stored in the history buffer. It enables you to recall specific previously entered commands. Activate line recall by typing a colon (:) followed by one or more characters that unambiguously abbreviate the command.

You can issue a number of special commands that interact with line recall at the command prompt. These commands and any defined user aliases (see below) can only be used at the command line interface. They are not an extension to ObjectScript and cannot be used in routines and methods.

The :? displays a quick help:

```
USER>:?
:<number>    Recall command # <number>
:alias       Create/display aliases
:clear       Clear history buffer
:history     Display command history
:unalias     Remove aliases
```

The :history command produces a numbered list of commands stored in the history buffer. You can use these numbers to recall a particular command, as shown in the following example:

```
USER>:history
1: zn "$SYS"
2: D ^SYSLOG
3: w "Hello"," World!"
4: w $J
5: :h
$SYS>:3
w "Hello"," World!"
Hello World!
```

The :alias command creates shortcuts for ObjectScript commands commonly typed at the terminal prompt. In addition to creating a short synonym for a long command, the alias definition accepts placeholder variables like $1 and $2 that are replaced by the actual arguments when the alias is expanded. The name of an alias is case-insensitive.

For example, one commonly typed line is:

```
USER>d $system.OBJ.Load("/somepath/myclass.xml","ck")
```

You can abbreviate this command with the following alias:

```
USER>:alias load Do $system.OBJ.Load($1,$2)
```

To invoke the alias, type:
or if there are no other commands or aliases starting with 'l' in the command history:

```
USER>:l "/somepath/myclass.xml" "ck"
```

It would be even simpler to add the quotes to the definition instead of at each invocation:

```
USER>:alias load Do $system.OBJ.Load("$1","$2")
USER>:l /somepath/myclass.xml ck
```

When an argument is not present in the invocation, its placeholder is skipped and doesn't generate any text, so:

```
USER>:l /somepath/myclass.xml
```

expands to

```
Do $system.OBJ.Load("/somepath/myclass.xml","")
```

You can use this feature to define aliases that support default arguments. For example, you might want to pass "ck" as the second argument to $System.OBJ.Load() when the corresponding argument is not specified in the alias invocation. You can achieve this effect with the following definition:

```
USER>:alias load Do $system.OBJ.Load("$1","$2")
```

Invoking `:alias` with a single argument displays the current definition assigned to that name. Note that both `:alias` and `load` are abbreviated.

```
USER>:a l
    load Do $System.OBJ.Load("$1","$2")
```

Invoking `:alias` without arguments displays all the defined aliases:

```
USER>:a
    load Do $System.OBJ.Load("$1","$2")
```

The command `:unalias <name>` removes the definition associated with that name.

The `:clear` command deletes all the contents from the recall buffer, but does not delete ~/.iris_history.

On Unix systems, if a file named ~/.iris_init exists, it is read when a terminal session is initiated. Only lines starting with `:<command>` are processed. All other lines are ignored.

Currently, this file accepts only `:alias` commands, which enables you to store favorite command abbreviations there.

Example of a ~/.iris_init file:

```
# Initial aliases
:alias % zn "%SYS"
:alias load Do $System.OBJ.Load("$1","$2")
:alias ut Do #$class(%UnitTest.Manager).RunTest("$1")
```

## 2.4.2 Reverse Incremental Search

Reverse incremental search (RIS) is an additional mode of operation of the command line recall facility. You enter this mode by typing Ctrl+R. In order to use RIS mode, there must be commands stored in the recall buffer. If the buffer is empty, the terminal bell rings and nothing else happens. Two backslash characters after the prompt indicate RIS mode:

```
USER>\\
```
The search string appears between the two backslashes, and the matching command, if any, appears to the right. As you type characters in the search string, the system scans the command history buffer in the same direction as typing up-arrow, from most recent to least recent. It stops at the first command that contains the string. Press Enter to execute the command.

For example, if the command \$J is in the recall buffer, after typing the 'S' key, the terminal might look like this:

```
USER>\$\ W \$J
```

If you want to find the command \$ZV instead, type the following character, Z:

```
USER>\$Z\ W \$ZV
```

On terminals that support ANSI attributes, the matching characters in the recalled command will be displayed with the bold attribute.

You exit the RIS mode when you type:

- **Enter** – selects and immediately executes the command.
- **Ctrl+G** – cancels RIS mode and goes back to the previous prompt without any command.
- An editing key, like the left arrow or **Ctrl+A** – this enters edit mode and allows you to edit the recalled command line.
- Up or down arrow – selects the previous or next command relative to the one that had been selected by the search.

You continue in the search mode when you type one of the following:

- A non control character – the character is appended to the search string and the system tries to find the previous command that contains the new string.
- **Ctrl+R** – searches for the previous command containing the same search string.
- **DEL** or **BS** – erases the rightmost character of the search string and takes you back to the corresponding command.

### 2.5 Copying and Pasting Text

You can copy and paste text in the Terminal. To do this, you can use the context menu (right-click menu), the Edit menu, or various keyboard shortcuts. The following options are available:

- **Copy** copies the selected text to the clipboard.
- **Paste** pastes the contents of the clipboard, line by line, to the current position of the cursor (which is the end of the Terminal scrollback buffer). The text becomes visible in the Terminal window unless echoing has been disabled.
- **Copy + Paste** copies the selected text to the clipboard and then pastes it, line by line, to the current location of the cursor.

#### 2.5.1 Keyboard Shortcuts

You can use the following keyboard shortcuts:

**Table 2–2: Shortcuts for Copy and Paste**

<table>
<thead>
<tr>
<th>Action</th>
<th>Basic Shortcut</th>
<th>Windows Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Ctrl+Ins</td>
<td>Ctrl+C</td>
</tr>
<tr>
<td>Paste</td>
<td>Shift+Ins</td>
<td>Ctrl+V</td>
</tr>
<tr>
<td>Copy and paste</td>
<td></td>
<td>Ctrl+Shift+V</td>
</tr>
</tbody>
</table>
The shortcuts listed in the “Basic Shortcut” column are always enabled.

The shortcuts listed in the “Windows Shortcut” column are enabled only if you set the Windows edit accelerators option to Yes; for information on this setting, see the section “User Settings,” in the chapter “Controlling the Appearance and Behavior of the Terminal.”

### 2.5.2 Notes about Copying and Pasting

- As noted above, if you set the Windows edit accelerators option to Yes, Ctrl+C copies the selected text to the Windows clipboard. To interrupt the Terminal, you must instead press Ctrl+Shift+C.
- If the host has a mouse request outstanding and you wish to do a local cut and paste, press Ctrl while selecting the region; that mouse action is not reported to the host.
- If the copied text includes a line boundary, it is saved on the clipboard as a carriage return and a line feed. If you do not want to paste line feeds, see “User Settings.”
- The Terminal can often paste data faster than a host can accept it. See “User Settings” for settings to control the speed of pasting. Also, line feeds can be discarded during a paste command.

### 2.6 Printing

To print from the Terminal, use the following options of the File menu:

- To select a printer and set it up for use with the Terminal, select File > Printer Setup.
- To print the contents of the Terminal screen, select File > Print.
- To print the log file (or any other ASCII file), select File > Print Log. This option lets you select the file to print, and it does no special processing except to try to be reasonable in processing form feed characters. During printing, mouse and keyboard input is locked out of the main window and a cancel dialog box appears. Printing is done in draft mode.

### 2.7 Clearing the Screen

The Edit menu provides two different options to clear the screen:

- To reset the screen, select Edit > Reset. This option resets the margins, scroll region and other processing on the current page, and causes the Terminal to repaint the window.
- To reinitialize the screen, select Edit > Erase or press Ctrl+Del. This option reinitializes the Terminal window, erases all session data, and resets the scrollbar region to zero.

### 2.8 Logging the Terminal Session

To start logging the Terminal session:

1. Select File > Logging or select Alt+L.
The Terminal displays a dialog box to prompt you for the location and name of the log file. The default directory is `install-dir/mgr`. The default filename is `TERMINAL.LOG`.

The total length of the path and file name cannot exceed 126 characters.

2. Optionally specify a different directory and filename.

3. Select **OK**.

   If the log file exists, the Terminal asks if you want to overwrite it and presents three choices:
   
   • **Yes** overwrites the file with the new log data
   
   • **No** appends any new log data to the file
   
   • **Cancel** leaves the file as is (no logging is done)

Later, to stop logging, select **File > Logging** or select **Alt+L**. The Terminal displays a dialog box to indicate that the log file is closed; select **OK**.

The log file contains only the output from a connection (independent of the current wrap mode).

You can also perform logging from a Terminal script, as described later in this book. Note that if you have started logging by using **File > Logging**, you cannot start a script that also performs logging. If you attempt to do so, the behavior is indeterminate.

Also see the section “**Learning Mode**” in the chapter “**Advanced Topics.**”
Controlling the Appearance and Behavior of the Terminal

This chapter describes different ways to control the appearance and behavior of the Terminal in interactive mode.

**Tip:** In addition to the options listed here, you can use the configuration option *TerminalPrompt*, which controls the Terminal prompt. This option is available in the Management Portal. To find it, select *System Administration > Configuration > Additional Settings > Startup*. For example, the Terminal prompt can include the configuration name of your system.

### 3.1 Specifying the Font

To specify the font size to use, select *Edit > Font*. This displays a dialog box where you can select a typeface, size, and style appropriate to your monitor and resolution.

**Note:** If you choose a font size that would expand the window beyond the borders of your screen, the Terminal automatically resizes both screen and font to the largest size available.

Also, whenever you switch to a different size screen, the Terminal attempts to use the preselected font for that size.

### 3.2 Specifying the Colors

To specify the colors to use, select *Edit > Color*. This displays a dialog box where you can select the default foreground and background colors for the Terminal. Then select one of the following:

- **Apply** changes only the current Terminal.
- **Save** does not change the current instance but saves the color information for new instances of the Terminal.

You can adjust the colors from the expected ANSI-named colors to any colors that the display board can deliver. These colors are saved along with the foreground and background choices. To select the default colors, select **Default** and then choose the colors.
3.3 Specifying the Window Size

To specify the Terminal window size, select **Edit > Window Size**. This displays a dialog box where you specify the window size:

The maximum number of columns is 132 and the maximum number of rows is 64. As you make changes, the dialog box updates the number of scrollback lines and the number of pages available. Then select one of the following:

- **Apply** changes only the current Terminal.
- **Save** does not change the current instance but saves the window size information for new instances of the Terminal.

**Note:** When you change the window size, the Terminal erases all current data in both the current display page and all back pages. Further, if there is a font selected for that size, the Terminal selects it.

3.4 Defining Custom Key Combinations

To define custom key combinations, select **Edit > User Keys**. This displays a dialog box where you can associate an ObjectScript command with any of the following key combinations: **Alt+Shift+F1** through **Alt+Shift+F10**.

Selecting **Ok+Save** updates the current instance and saves the key sequences for future instances of the Terminal.

To include a non-printable characters in the command, use the decimal equivalent (**nnn**) of the character. You may also use one of **<CR>**, **<F10>**, **<F7>**, **<DO>**, **<TAB>**, **<LF>**, **<ESC>**, **<CSI>**, **<NL>** (= **<CR><LF>**).

You can also use **<P1>**, **<P2>**, and other command line parameters.

**Note:** There are known problems with the User Keys facility. For up-to-date information, please contact the InterSystems Worldwide Response Center (WRC).

3.5 Specifying Other User Settings

To specify user settings, select **Edit > User Settings**. This displays a dialog box where you can specify both the current setup and initial values of various parameters used by the Terminal. The settings are as follows:
### Table 3–1: User Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrap</td>
<td>Automatic wrapping at right hand column</td>
</tr>
<tr>
<td>&lt;X&gt; Key sends ^H</td>
<td>Set to have &lt;X&gt; key send Ctrl+H rather than Delete</td>
</tr>
<tr>
<td>Application Keypad</td>
<td>Enable Application Mode keypad</td>
</tr>
<tr>
<td>Force Numeric Pad</td>
<td>Force standard PC keypad</td>
</tr>
<tr>
<td>Disable Special ID</td>
<td>Do not send special Terminal ID</td>
</tr>
<tr>
<td>Disable Mouse Reports</td>
<td>Do not send mouse reports</td>
</tr>
<tr>
<td>Enable Fast Paint</td>
<td>Enable fast paint mode</td>
</tr>
<tr>
<td>Paste Keeps Linefeed</td>
<td>Set to send line feeds from clipboard (along with carriage return)</td>
</tr>
<tr>
<td>Pass XOFF Through</td>
<td>Let remote host handle XOFF/XON.</td>
</tr>
<tr>
<td>Windows edit accelerators</td>
<td>Specifies whether the Terminal enables the common Windows edit shortcuts</td>
</tr>
<tr>
<td></td>
<td>(Ctrl+C, Ctrl+V, Ctrl+Shift+V), in addition to the basic Terminal edit</td>
</tr>
<tr>
<td></td>
<td>shortcuts (Ctrl+Insert and Shift+Insert).</td>
</tr>
<tr>
<td></td>
<td>If these Windows shortcuts are not enabled, the characters are passed in the</td>
</tr>
<tr>
<td></td>
<td>data stream to the InterSystems IRIS server process.</td>
</tr>
<tr>
<td>Paste burst size (in bytes)</td>
<td>The number of characters to paste in one transmission</td>
</tr>
<tr>
<td>Paste pause time (in msec)</td>
<td>The number of milliseconds between successive transmissions of pasted</td>
</tr>
<tr>
<td></td>
<td>material longer than the burst size</td>
</tr>
</tbody>
</table>

Some systems cannot accept data as fast as the Terminal can send it. Thus, the **Paste burst size** determines how much to send at once and the **Paste pause time** determines the pause time between transmissions. If either setting is less than 1, the entire clipboard is sent at once.

### 3.6 Specifying the Network Encoding

The network encoding of the Terminal controls how characters are translated at the following times:

- When the Terminal translates keyboard input into its display memory, which uses Unicode. Characters received from the keyboard are translated from the single or multibyte character stream using the current Windows input character set. This means that if you change your input language, the Terminal recognizes the change and adapts to it. This enables you to type in mixed languages; the Terminal recognizes it and translates properly into its internal Unicode representation. If you use mixed-language input, select **UTF8** as your network encoding and your InterSystems IRIS $ZMODE I/O translation table.

- When the Terminal communicates with a peer server. Characters transmitted to the server are translated from the internal Unicode representation to a network encoding and characters received from the server are translated from the network encoding to Unicode.

The default network encoding is UTF8.
To specify the network encoding, select Edit > Network Encoding. This displays a dialog box where you can choose the network encoding for the Terminal to use. There are 4 choices: UTF8, Windows, ISO, and EUC. Because these encodings are not all relevant to every input locale, only the relevant choices are enabled on the menu.

### 3.6.1 UTF8 Encoding

When you select the UTF8 option, the Terminal translates the internal Unicode characters to UTF8 on output to the server and from UTF8 when received from the server. If you select UTF8, the InterSystems IRIS I/O translation for your principal I/O device must be UTF8. You can determine the I/O translation from $ZMODE. It is the fourth field; fields are delimited by backslashes (\).

### 3.6.2 Windows Encoding

When you select the Windows option, the Terminal uses the current Windows input code page to translate I/O between the Terminal and the server, to and from the internal Unicode character set encoding. When you use the Windows encoding, make sure to set the InterSystems IRIS I/O translation ($ZMODE) to that it expects the character set represented by the active Windows code page.

### 3.6.3 ISO Encoding

When you select the ISO option, the Terminal uses the following ISO 8859-X code pages to translate I/O to and from the peer server. The Terminal selects the appropriate ISO code page based on the current Windows input code page. The following mappings are enabled:

#### Table 3–2: ISO Encoding

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Western European</td>
<td>8859-15</td>
<td>1252</td>
<td>28605</td>
</tr>
<tr>
<td>Central European</td>
<td>8859-2</td>
<td>1250</td>
<td>28592</td>
</tr>
<tr>
<td>Cyrillic</td>
<td>8859-1</td>
<td>1251</td>
<td>28591</td>
</tr>
<tr>
<td>Greek</td>
<td>8859-7</td>
<td>1253</td>
<td>28597</td>
</tr>
<tr>
<td>Turkish</td>
<td>8859-9</td>
<td>1254</td>
<td>28599</td>
</tr>
<tr>
<td>Hebrew</td>
<td>8859-8</td>
<td>1255</td>
<td>28598</td>
</tr>
<tr>
<td>Arabic</td>
<td>8859-6</td>
<td>1256</td>
<td>28596</td>
</tr>
<tr>
<td>Baltic Rim</td>
<td>8859-4</td>
<td>1257</td>
<td>28594</td>
</tr>
<tr>
<td>Korea</td>
<td>iso-2022-kr</td>
<td>949</td>
<td>50225</td>
</tr>
<tr>
<td>Japan (JIS)</td>
<td>N/A</td>
<td>932</td>
<td>50220</td>
</tr>
</tbody>
</table>

All other Windows input code pages use the Windows code page if the ISO network encoding is selected.

When using the ISO encoding, you must ensure that the InterSystems IRIS I/O translation shown in $ZMODE is consistent with the character set represented by the active ISO code page used by the Terminal.
### 3.6.4 EUC Encoding

The EUC encoding is relevant to far Eastern languages and is used to communicate with certain UNIX® systems. When you select the EUC option, the Terminal uses the following code pages to translate I/O to and from the server. The Terminal selects the appropriate EUC code page based on the current Windows input code page. The following mappings are enabled:

**Table 3–3: EUC Encoding**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>N/A</td>
<td>932</td>
<td>51932</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>N/A</td>
<td>936</td>
<td>51936</td>
</tr>
<tr>
<td>Korean</td>
<td>N/A</td>
<td>949</td>
<td>51949</td>
</tr>
</tbody>
</table>

Japanese (JIS) support is provided under the ISO network encoding using the 50220 code page to translate to/from the internal Unicode.

### 3.7 Specifying the Physical Character Setting of the Display

To specify the physical character setting, select **Edit > Display Physical Character Setting** and then select either **Logical** and **Physical**. This option controls the aspect of the characters displayed in the Terminal window. The difference is apparent only when using multibyte character sets.
This chapter discusses how to create and use script files for the Terminal.

Also see the next chapter, “Script Command Reference” and see the section “Learning Mode” in the chapter “Advanced Topics.”

Note: Terminal scripts are useful on occasion, but it is usually much easier to write and use a routine, because each of the routine programming languages provides a much richer set of options.

The user environment variables and shared drive letter designations are those defined by the user account in which the InterSystems IRIS control service runs.

### 4.1 Starting Script Execution

Script files (with default extension .scr) are normally found in the working directory but could be anywhere.

To run a script, select **File > Script** or press **Alt+S**. A standard Windows file lookup box is presented and the selection of a script is made.

If a script is given as an argument on a command line, it is started immediately if no switch is locking the command mode, or deferred until after a host connection is made if there is a switch.

Note: If you edit the communications choices to a single mode, that is equivalent to locking the Terminal to a single choice and thus any script file is invoked after the host connection is made.

### 4.2 Pausing Script Execution

To pause the execution of a script, select **File > Pause** or press **Alt+P**. You are prompted to confirm that you want to pause the current script.

### 4.3 Stopping Script Execution

To stop a script, select **File > Script** or press **Alt+S**. You are prompted to confirm that you want to stop the current script.
4.4 Contents of a Script File

Script files are line oriented; there is no line-continuation convention. Each line is separate from any other. Lines beginning with a semicolon are considered comments. You can use blank lines liberally to improve readability. Normally, invalid lines are ignored. Script commands may be preceded by spaces and/or tabs.

The format for a line that contains a script command is as follows. Note that arguments are interpreted as strings or numbers:

\[ \text{ScriptCommand}: \text{ScriptArguments} \]

Here \textit{ScriptCommand} is one of the Terminal script commands and \textit{ScriptArguments} is the argument list for that command (see details for the specific commands). Note that if script command consists of two or more words, the words of the command must be separated from each other by a single space. Also note that there is no space between the command and the colon.

Or, for a command that has no arguments:

\[ \text{ScriptCommand} \]

You can use labels to define points of control transfer. A label begins with a dollar sign ($), is not case-sensitive, and can have embedded spaces. A label must appear by itself on a line.

Also see the section “Learning Mode” in the chapter “Advanced Topics.”

4.5 Script Command Summary

The following table gives the list of available script commands:

\textit{Table 4–1: Terminal Script Commands}

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>Transmit a break for those communications devices that support it</td>
</tr>
<tr>
<td>call script</td>
<td>Exit the current script and start another</td>
</tr>
<tr>
<td>case match</td>
<td>Indicate if the &quot;wait for&quot; string must match in case</td>
</tr>
<tr>
<td>closelog</td>
<td>Close a log file</td>
</tr>
<tr>
<td>connect</td>
<td>Force a host connection if not connected</td>
</tr>
<tr>
<td>debug</td>
<td>Enable/disable debugging for scripts</td>
</tr>
<tr>
<td>disconnect</td>
<td>Force a disconnect if connected</td>
</tr>
<tr>
<td>display</td>
<td>Send text to the display</td>
</tr>
<tr>
<td>echo</td>
<td>Turn on/off echo of incoming characters</td>
</tr>
<tr>
<td>execute</td>
<td>Execute a Windows program</td>
</tr>
<tr>
<td>exit</td>
<td>Exit the script</td>
</tr>
<tr>
<td>goto</td>
<td>Transfer control to another place in the script</td>
</tr>
<tr>
<td>if empty</td>
<td>Transfer control if last test string was empty</td>
</tr>
<tr>
<td>Command</td>
<td>Action</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>key_startime</td>
<td>Simulate key timing start</td>
</tr>
<tr>
<td>key_stoptime</td>
<td>Simulate key timing stop</td>
</tr>
<tr>
<td>key_timer</td>
<td>Turn key timing on and off</td>
</tr>
<tr>
<td>logfile</td>
<td>Start a log file</td>
</tr>
<tr>
<td>multiwait for</td>
<td>Wait for any of several strings from the communications device</td>
</tr>
<tr>
<td>notify</td>
<td>Display a dialog box and wait for user response</td>
</tr>
<tr>
<td>on error</td>
<td>Indicate label to branch to if timer fires</td>
</tr>
<tr>
<td>pause</td>
<td>Pause the script</td>
</tr>
<tr>
<td>return</td>
<td>Return from a subroutine in the script file</td>
</tr>
<tr>
<td>send</td>
<td>Send text to the communications device</td>
</tr>
<tr>
<td>subroutine</td>
<td>Call a subroutine in the script file</td>
</tr>
<tr>
<td>terminate</td>
<td>Exit the emulator entirely</td>
</tr>
<tr>
<td>test</td>
<td>Build a string to be tested</td>
</tr>
<tr>
<td>timer</td>
<td>Control the timer for &quot;wait for&quot;</td>
</tr>
<tr>
<td>title</td>
<td>Set the window title</td>
</tr>
<tr>
<td>wait for</td>
<td>Wait for a particular string from the communications device</td>
</tr>
</tbody>
</table>

For reference information on these commands, see the later section “Script Command Reference.”

### 4.6 Script Command Arguments

All spaces and tabs at the beginning and end of arguments are ignored.

All numeric arguments are integers. A required numeric argument defaults to 0 if not provided. Additionally, OFF is equivalent to 0 and ON is equivalent to 1.

Strings are simply the concatenation of all data on a line after the command (with the exception of the leading and trailing white space). Quotation marks are not needed. Additionally, parameter substitution is accomplished with the use of one of the following:

\(<P1>, <P2>, ..., <Pn>\)

This substitutes the \(n\)-th command line parameter in place of \(<Pn>\).

To ease operation, certain ASCII characters have equivalent shortcut representations as shown in the following table.

**Note:** Any ASCII (extended) character except NUL (000) can be produced via \(<ddd>\) where \(ddd\) is the decimal value of the character.

#### Table 4–2: Special Characters

<table>
<thead>
<tr>
<th>Character(s)</th>
<th>Interpretation</th>
<th>Transmitted Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;CR&gt;</td>
<td>Carriage return</td>
<td>&lt;13&gt;</td>
</tr>
</tbody>
</table>
### 4.7 A Sample Script

The following is a sample Terminal script:

```plaintext
; initialization -- turn match off to make comparisons more lenient
case match: off

; wait for the terminal to initialize and ask for our identification
echo: off
wait for: Username
send: SYS<CR>
wait for: Password
send: XXX<CR>
title: Terminal Example
echo: on

; log everything in a log
logfile: C:\TermExample.log
; wait a second
pause:10

; display a header to let the user know we are ready
; you need <CR><LF> because "display" does not
; have a prompt to advance to another line
display::<CR><LF>
display:--------------------------<CR><LF>
display:<<< Terminal Example >>><CR><LF>
display:--------------------------<CR><LF>
; wait a second
pause:10

; switch to the USER namespace
send: set $namespace="USER"<CR>
wait for: USER>

; display some basic information about the system
; Use the debugging routine to do so
send: Do ^%STACK<CR>
wait for: action:

; have it outline our options
send: ?<CR>
wait for: action:

; wait 5 seconds for user to absorb
pause: 50

; ask for the basic process info
```
send: *s
pause: 50
send: <CR>
wait for: action:
; wait another 10 seconds
pause: 100
; finish the session
send: <CR>
; close the log file
closelog
; finished
terminate
This chapter provides reference information for the Terminal script commands.

**Note:** Terminal scripts are useful on occasion, but it is usually much easier to write and use a routine, because each of the routine programming languages provides a much richer set of options.

### 5.1 break

Sends a break for those communications nodes that support a break. Otherwise, it does nothing. It has no arguments. Usage example:

```bash
break
```

### 5.2 call script

Starts running a script. If a script is executing when this command is executed, the Terminal terminates that script before starting the new script. Usage example:

```bash
call script: login fred <p3>
```

This example stops the current script (if one is running) and starts another called `login.scr`. The first parameter in the sample script (`login.scr`) is `fred`. The second parameter is whatever is in the third parameter of the current script file (the one making the call). The default file extension is assumed to be `.scr`. The current working directory is searched first for an instance of `login.scr`.

### 5.3 case match

Enables or disables exact case matching in the `wait for` command. Usage example:

```bash
case match: off
```

By disabling exact case matching, you can match strings even if the individual characters differ from one another in case. The default for this switch is `on`. 
5.4 closeLog

Closes the currently open log file, if any. Usage example:

    logfile: mydirect.log
    send: dir *.*/FULL<CR>
    wait for: <NL>$
    closeLog

5.5 connect

Opens a dialog box to initiate a connection to a remote host. Usage example:

    connect

5.6 debug

Enables debug mode, which traps invalid script commands, which the Terminal usually ignores. When you enable the debug mode, the first part of an invalid command appears in a message box requiring operator attention. Usage example:

    debug: on

5.7 disconnect

Disconnects from the host. If the Terminal is not connected, the command does nothing. Usage example:

    disconnect

5.8 display

Writes data to your screen. It is not sent to the communications device. Usage example:

    display: <CSI>H<CSI>J<LF>Here are the choices for today:

When this example is executed, it moves the cursor to the home position, clears the window, advances by one line, and writes the text Here are the choices for today: and leaves the cursor after the end of the text.

5.9 echo

Enables or disables displayed output to the window and log file. This is useful if you need to hide output (because it is uninformative to the user, for example). For an example, see wait for.
5.10 execute

Launches a Windows program with a SHOW attribute for its window. Usage example:

```
execute: notepad.exe myfile.not
```

This example executes the Windows Notepad program and opens the file `myfile.not` inside that application. Notice that you could do the following:

```
logfile: mydat.lst
echo: off
send: dir *.dat/full
wait for: <NL>$
closelog
echo: on
execute: notepad mydat.lst
```

**Note:** No test is made to see if the program actually starts and no wait is done for its completion.

5.11 exit

Exits the script. A script normally exits when it reaches the end of its last line, but you may wish to exit if some event (say a login) does not occur. Usage example:

```
on error: $byebye
timer: 40
wait for: event:
goto: $Got event
$byebye:
    notify: Did not find event prompt, exiting script
    exit
$Got event:
    timer: 0
    ; more commands
```

5.12 goto

Transfers control to another place in the script file. This is useful for managing control flow for looping, and in response to timeout branching. Usage example:

```
on error: $Not There
timer: 30
wait for: abc<CR>
goto: $Got It
$Not There:
    ; failed to see it, send Ctrl+C
    send: <3>
goto: $bad
$Got It:
    ; turn timer off because we got abc<CR>
timer: 0
    ; more commands ...
5.13 if empty

Causes a branch to the given label if the last test command found an empty string. Usage example:

```
  test: <p1>
  if empty: $No First Arg
```

The first command determines if the first parameter provided in the command line is missing or empty. The second command branches to the label $No First Arg if this is the case.

5.14 key_starttime

Starts the timing of a key sequence. It takes a single numeric argument. If the argument is zero, the statistics are accumulated when you press Enter. Otherwise, statistics are accumulated when you press F10. Usage example:

```
  key_starttime: 0
```

To stop timing, use the key_stoptime command.

5.15 key_stoptime

Stops a timing and accumulates statistics, if timing is currently active. Usage example:

```
  key_starttime: 0
  wait for: <esc>[14;22H
  key_stoptime
```

5.16 key_timer

Starts or stops the data collection of key timing information. Alternatively, you can start or stop the timer with Alt+Shift+T. Usage example:

```
  key_timer: on
  ; rest of your script commands
  key_timer: off
```

A file (KEYTIMER.LOG) is constructed in the system manager’s directory that contains a histogram of key timings. You can use only one such timing sequence because it does not append to the current statistics file but overwrites it.

**Note:** To drive timings exclusively from a script file, you must use <CR> and <F10> in place of <13> and “<27>[21-”, respectively.
5.17 logfile

Starts the collection of received data in the log file specified. If there is currently a log file active, it is quietly stopped. Use the `closelog` command to stop the logging. Usage example:

```
logfile: mydirect.log
send: dir *./*FULL<CR>
wait for: <NL>$
closelog
```

The default directory is the directory in which the script resides. It can be changed by supplying a full pathname. Log files are normally opened for overwrite; that is, if they already exist, new data replaces what is already there.

5.18 multiwait for

Synchronizes the script file with the host. Processing is suspended until the data that arrives from the host matches one of several strings given in the argument. Usage example:

```
multiwait for: =USER>=***ERROR,=DONE
```

This example causes the script file to wait (maybe forever) until any one of three strings of characters arrives. The first non-blank character of the argument (in this instance, the equals sign) is treated as the delimiter that breaks up the argument into substrings. So this command causes the example script to wait until one of the following sequences arrives:

```
USER>
***ERROR,
DONE
```

You can use a timer to break out of this command.

See the `case match` command to turn exact case matching on or off.

Because a `case match` command may have only a single substring argument, the following two script commands are identical in function:

```
multiwait for: =USER>
wait for: USER>
```

5.19 notify

Displays a Windows message box and waits until the user presses OK. You can use this for messages to the user who runs the script. Usage example:

```
notify: Ready to send commands...
send: copy *.lst backup:*.lst<CR>
send: delete *.lst;
```

Note: This box is modal and cannot be interrupted except by the user.
5.20 on error

Establishes the target label for the implied goto that is executed if a timer expires (normally while waiting for text to arrive). To be strictly correct, you should use this command before using timer, although in practice the order might not matter.

For examples, see wait for, exit, goto, and subroutine.

Also see the timer command.

5.21 pause

Pauses a running script for a number of tenths of seconds. Usage example:

    pause: 30

This example pauses execution of the script for three seconds. If the argument is 0, that is equivalent to an indefinite pause; to resume from an indefinite pause, use Alt+P.

5.22 return

Used with the subroutine command to return to the place in the script from which the subroutine was called. See the subroutine command for an example.

5.23 send

Simulates typed data that is sent to the currently connected host. Usage example:

    send: set $namespace="%SYS"<CR>

This line changes the namespace to %SYS. The <CR> at the end is necessary because the send command does not implicitly add a carriage return.

Another usage example is as follows:

    send: 1<cr>2<cr>A1234<F10><32>

This command is equivalent to typing the following, in sequence:

- The character 1
- The carriage-return key
- The character 2
- The carriage-return key
- The string of characters A1234
- The F10 key
- A single space character
Notice that `<32>` is the only way to send a leading or trailing space. Those characters are removed by the command interpreter if typed normally.

**Important:** It is best practice to include the `wait for` command after each `send` command. The `wait for` command provides the ability to synchronize later commands in the script with input from the terminal. The Terminal script mechanism sends commands one-after-the-next without regard to the input returning from InterSystems IRIS except when a `wait for` command is encountered.

If you do not include the `wait for` command after each `send` command, and you are generating a log file, the log file will not include information for the later `send` commands.

### 5.24 subroutine

Useful if repetitive commands are used in a script. It saves both memory and the possible need to invent many different labels. Use this command with a `return` command. Usage example:

```plaintext
subroutine: $Send It Again
; some other processing
exit

$Send It Again:
send: <F7>Q
on error: $skip
timer: 30
wait for: [22;5H
timer: 0
return

$skip:
send: <3>
; note on error still set to $skip
timer: 30
wait for: function:
timer: 0
send: <CR>
exiit
```

**Note:** The subroutine stack holds 16 addresses. If you try to nest subroutine invocations deeper than that, the script fails.

### 5.25 terminate

Tells the Terminal to exit back to Windows. Any open files are closed, extra windows are removed, and the connections are closed. Usage example:

```plaintext
terminate
```

### 5.26 test

Tests if a parameter or window property is non-empty. The command is used in conjunction with the `if empty` command. See the `if empty` command for an example.
5.27 timer

Sets a timer to use with the wait for command. The timer command performs a Windows SetTimer() command. When the timer fires, the script processor goes to the label specified by on error (if any). The script exits immediately if no label has been specified by on error.

Usage example:

```
timer: 100
```

The argument is the number of tenths of a second to wait. The example sets a timer for ten seconds. See the example in the goto command for another example.

To switch off a timer, use the following:

```
timer: 0
```

For an example of timer in context, see wait for.

5.28 title

Sets the Terminal window title to the specified string. Usage example:

```
title: This is my window
```

You can also set the title remotely via the extended emulator commands.

5.29 wait for

Synchronizes the script file with data that arrives from the host. Usage example:

```
wait for:  USER>
```

This example causes the script file to wait (maybe forever) until the precise string of characters USER> arrives. This particular sequence is the default prompt from the Terminal when in the USER namespace. This means that you can use this command to wait until the Terminal is ready for more input.

You can use timer to break out of a wait for command. If the text being looked for is never received, or is missed due to timing or case match issues, timer is the only way to interrupt a wait for and continue execution of the script. When you use timer, you can specify a label to receive the flow if the timer expires; see on error. If wait for finds the text it is seeking, it kills the timer and clears any label set by on error. Usage example:

```
echo: off
on error: $Failed Login
timer: 50
wait for: Name:
send: <p1><CR>
wait for: Password:
send: <p2><CR>
wait for: <NL>$
echo: on
Notify: Login is complete
display: <CSI>H<CSI>J
send: <CR>
goto $Process
```

$Failed Login:
    echo: on
    notify: Login failed.
    exit

$Process:
    ;processing begins

This example hides the login sequence using a name and password supplied as the first two script parameters, respectively. If the login succeeds, processing begins at the indicated label. If it fails, the script displays a dialog box indicating the failure and then exits when you select OK.

The `wait for` command may or may not consider the case of the text, depending on whether and how you have used the `case match` command.
Connecting to Remote Hosts

To connect the current Terminal session to a database on a remote host, use the Connect menu.

- To connect to a remote host, select Connect > Host or press Alt+O. This brings up a dialog box where you can enter the address of the desired host.
  
  Or select the name of the host from the Connect menu.
- To send a break over the communications channel, select Connect > Send Break or press Alt+B.
- To disconnect or to cancel a connection attempt, Connect > Disconnect.

Note: If you start the Terminal with either the /console or the /server control argument, the Connect menu item is not shown.

6.1 Example

This example starts an instance of the Terminal and then manually connects it to the TELNET port on the local host to enable a console session. The example assumes that the default user ID and password are available.

1. Select Connect > Host.
2. In the dialog box that appears, enter 127.0.0.1 for the Remote System address and 23 for the Port Number. Select OK.
   
   The Terminal application then attempts to connect to your local host via the TELNET port.
3. At the Username: prompt, enter SYS and press Enter.
4. Then, at the Password: prompt, enter a password and press Enter.
   
   You then see a prompt (%SYS>) that indicates that you are connected and have been placed in the %SYS namespace.
5. Enter ObjectScript commands as needed.
6. To terminate the session, select Connect > Disconnect. Or, to terminate this Terminal, select the Close box in the upper right of the window.
7

Using the Terminal in Batch Mode

For some operating systems, you can run the Terminal from the command line (for example, the DOS window).

7.1 The Batch Command Line

On Windows systems, you can invoke the Terminal from the DOS command line (cmd.exe, to be precise). The general form of the command line is:

```
iristerm Arg1 Arg2 ... ArgN ScriptFilePath
```

Where:

**Table 7–1: Invoking the Terminal in Batch Mode**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>iristerm</td>
<td>Invokes the Terminal application. If the Windows environment variable <em>PATH</em> includes the location of the InterSystems IRIS binaries, then use the command name <code>iristerm</code> or <code>iristerm.exe</code>. Otherwise, you must use a full or partial path. For a default installation of InterSystems IRIS, the binaries are in the directory <code>install-dir\Bin</code>.</td>
</tr>
<tr>
<td>Arg1 ... ArgN</td>
<td>Control arguments (described in the next section).</td>
</tr>
<tr>
<td>ScriptFilePath</td>
<td>The location of the script file.</td>
</tr>
</tbody>
</table>

7.2 Control Arguments

Several arguments modify the starting environment for the Terminal session. Some of these are reserved for internal use and are not described here. The most useful arguments are as follows.

**Note:** If you start the Terminal with either the `/console` or the `/server` control argument, the Connect menu item is not shown.
7.2.1 /console=<ConnectString>

This argument specifies both the type of connection and the additional data needed to make the connection. There are two types: TELNET connections and connections to local console applications.

**Note:** You cannot specify both /console and /server arguments.

7.2.1.1 /console=cn_iptcp:<HostAddr>

This specifies the target system with which the Terminal is to interact over a TELNET connection. This is useful for running a script on the local machine. In this case, you specify the local machine IP address and port as HostAddr. For example:

```
iristerm /console=cn_iptcp:127.0.0.1[23]
```

7.2.1.2 /console=cn_ap:<Instance>[<Namespace>]

You can open a Terminal session. For example:

```
iristerm /console=cn_ap:iris[USER]
```

This line opens a Terminal session and switches to the given namespace.

In this case, the instance name is iris, and the namespace name is USER.

The namespace name is optional. If it is not supplied, the default namespace is used.

7.2.1.3 /console=cn_ap:<Instance>[<Namespace>]:<Routine>

You can also execute routines from a batch file. For example, assume that the following appears as a line in a batch (.bat) file for use in Windows 7:

```
iristerm /console=cn_ap:iris[USER]:^^%%D
```

This line starts the given instance (if it is not already running), opens a Terminal session, switches to the given namespace, executes the given routine. When the routine ends, the Terminal session is closed.

In this case, the instance name is iris, and the namespace name is USER. The routine name is ^%D (which prints out the current date).

The namespace name is optional. If it is not supplied, the default namespace is used.

7.2.2 /server=<ServerName>

This argument specifies the name of the server to use for a secure connection between this Terminal session and the given server.

```
iristerm /server=ServerName
```

For ServerName, specify an InterSystems IRIS server. To see the list of available servers, select the InterSystems Launcher and then select Preferred Server. The system then displays a list of servers.

Make sure of the following, on platforms other than UNIX®:

- On the desired server, the Telnet service (%Service_Telnet) must be enabled. (Note that this service is not enabled by default.)
  
  For information, see “Services” in the Security Administration Guide.
- The server must be running.
On UNIX®, the server does not need to be running, but you will log in to a shell, not directly in to InterSystems IRIS.

**Note:** You cannot specify both `/console` and `/server` arguments.

### 7.2.3 /size=RowsxCols

This argument gives the initial size of the Terminal screen, in terms of rows and columns. Each of $Rows$ and $Cols$ must be an unsigned integer. The $x$ that separates them is required as shown. No spaces are permitted in the control argument.

The allowed ranges for $Rows$ and $Cols$ are:
- $5 \leq Rows \leq 64$
- $20 \leq Cols \leq 132$

### 7.2.4 /pos=(X,Y)

This argument gives the initial origin of the Terminal screen in the display device window in pixels. Both $X$ and $Y$ must be unsigned integers. The parentheses around the pair and the comma separator are required. No spaces are permitted in the control argument.

**Note:** It is possible to place a window outside the displayed area by using values for $X$ and $Y$ that are larger than the size of the display device.

### 7.2.5 /ppos=(Xpct,Ypct)

This argument gives the initial origin of the Terminal screen in the display device window in terms of a percentage of the display area. Both $Xpct$ and $Ypct$ must be unsigned integers. The parentheses around the pair and the comma separator are required. No spaces are permitted in the control argument.

The allowed ranges for $Xpct$ and $Ypct$ are:
- $0 \leq Xpct \leq 40$
- $0 \leq Ypct \leq 40$

That is, the window origin cannot be placed above and to the left of the device origin. Nor can it be placed more than 40% down or across the display device.

### 7.2.6 /UnbufferedLogging

This argument causes output to be written immediately to the log file when logging is active instead of being buffered. This may be useful if another process is inspecting the output of the log file.

### 7.3 Examples

This section uses the example script shown earlier in this document, and shows two ways to run it.
### 7.3.1 Running a Script in Batch Mode

This example uses the basic debugging routine `^%STACK` to display information about the current user and the Terminal process. If the script commands are stored in `C:\TestScript.scr`, you can run the example by typing into a DOS command window:

```
C:\InterSystems\iris\bin\iristerm.exe /console=cn_iptcp:127.0.0.1[23] C:\TestScript.scr
```

### 7.3.2 Running a Script Interactively

You can invoke the previous routine in a Terminal session as follows:

```
C:\InterSystems\iris\bin\iristerm.exe /console=cn_ap:iris[USER]:^^%STACK
```

You can manually provide the responses that the script supplied in that example. When you reply to the prompt `Stack Display Action:` by pressing the `Enter` key, the Terminal window closes.
This chapter discusses a number of advanced topics.

### 8.1 Disabling the Close Button of the Terminal

If you need to disable the close button (X) of the Terminal, add a registry key, as follows:

- **On 32-bit Windows machines:**
  
  HKEY_LOCAL_MACHINE\SOFTWARE\InterSystems\Terminal\NoExit

  Notice the space in “Terminal”.

- **On 64-bit Windows machines:**
  
  HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\InterSystems\Terminal\NoExit=1

The NoExit value is of type REG_SZ in both cases.

### 8.2 Mapping of the Extended Keyboard

The Terminal supports application keyboard mode for the extended keyboard as follows:

**Table 8–1: Key Mappings**

<table>
<thead>
<tr>
<th>Key</th>
<th>Mapped Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num Lock</td>
<td>PF1</td>
</tr>
<tr>
<td>Keypad-divide</td>
<td>PF2</td>
</tr>
<tr>
<td>Keypad-times</td>
<td>PF3</td>
</tr>
<tr>
<td>Keypad-minus</td>
<td>PF4</td>
</tr>
<tr>
<td>Keypad-plus</td>
<td>Keypad-comma</td>
</tr>
<tr>
<td>Shift+Keypad-plus</td>
<td>Keypad-minus</td>
</tr>
<tr>
<td>F1, F2, F3, F4</td>
<td>PF1, PF2, PF3, PF4 (respectively)</td>
</tr>
<tr>
<td>Shift+F1 ... Shift+F10</td>
<td>F11 ... F20 (respectively)</td>
</tr>
</tbody>
</table>
The keypad cluster of the extended keyboard is mapped as follows:

**Table 8–2: Keyboard Mappings**

<table>
<thead>
<tr>
<th>Key</th>
<th>Mapped Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert</td>
<td>Insert Here</td>
</tr>
<tr>
<td>Home</td>
<td>Find</td>
</tr>
<tr>
<td>Page Up</td>
<td>Prev Screen</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove</td>
</tr>
<tr>
<td>End</td>
<td>Select</td>
</tr>
<tr>
<td>Page Down</td>
<td>Next Screen</td>
</tr>
</tbody>
</table>

The **Pause** key acts as a single XON/XOFF toggle key.

### 8.3 Special Modes

The Terminal provides a couple of special modes, accessible only by keyboard actions.

#### 8.3.1 Key Timing Mode

To enter or exit *key timing mode*, press Alt+Shift+T.

This mode can help you determine performance of a host system under various load conditions. The output of a timing run is the file KEYTIMER.LOG in the InterSystems IRIS system manager’s directory.

#### 8.3.2 Learning Mode

In *learning mode*, the Terminal generates a log file that you can quickly convert to a script file, after making only minor edits. When this mode is enabled, the log file is a sequence of script **wait for** and **send** commands. The **wait for** commands show up to 16 characters preceding the sent data.

To enter learning mode:

1. Press Alt+L to enable logging. Then specify a log filename and directory, as described in “Logging the Terminal Session,” earlier in this book.
2. Press Alt+Shift+L.

To exit learning mode, press Alt+Shift+L.

### 8.4 Using the Terminal with DDE

The Terminal supports DDE (Dynamic Data Exchange) links to permit other applications to talk to a remote host. This section assumes that you are familiar with DDE. The topics here are as follows:

- **Layout** — Used to obtain status information. Examples are the row and column size, whether there is a connection, and so on.
• **Screen** — Used to gather data from the Terminal screen.
• **Message** — Used to send data to either the Terminal screen or the host.

**Note:** A Windows task cannot discriminate between multiple instances of the Terminal when it comes to using DDE. Therefore, use DDE only if one copy of the Terminal is running.

### 8.4.1 DDE Layout Connections

The Terminal supports DDE requests for what could be considered as static information through the Layout topic.

**Table 8–3: DDE Layout Connections**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning of Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>The number of columns of the window.</td>
</tr>
<tr>
<td>Row</td>
<td>The number of rows of the window.</td>
</tr>
<tr>
<td>hWnd</td>
<td>The decimal equivalent of the main window handle.</td>
</tr>
<tr>
<td>Connected</td>
<td>A null string if there is no connection, otherwise the equivalent of the title string &quot;mode: node&quot;</td>
</tr>
<tr>
<td>Read</td>
<td>A 1 if the last received character was a CTRL/A. Its use is detection of the end of screen painting.</td>
</tr>
<tr>
<td>Script</td>
<td>A 1 if a script is currently running, otherwise 0.</td>
</tr>
<tr>
<td>Title</td>
<td>The title of the window.</td>
</tr>
</tbody>
</table>

### 8.4.2 DDE Screen Connections

The Terminal supports DDE requests for screen data through the Screen topic. Currently, one POKE command is available to select which part of a screen line is desired.

**Table 8–4: DDE Screen Connections**

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning of Returned Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cursor</td>
<td>The current position of the cursor, in the form row;col.</td>
</tr>
<tr>
<td>Line</td>
<td>The current line (without a CR LF).</td>
</tr>
<tr>
<td>LeftLine</td>
<td>The left part of the current line up to but not including the character under the cursor.</td>
</tr>
<tr>
<td>RightLine</td>
<td>The right part of the current line including the character under the cursor.</td>
</tr>
<tr>
<td>All</td>
<td>The entire screen, each line is delimited by CR LF.</td>
</tr>
<tr>
<td>Piece</td>
<td>The currently selected piece of a screen line (without a CR LF).</td>
</tr>
</tbody>
</table>

**Note:** The item "Piece" can be POKEd with a string of the form "RnnCmmLpp" to cause the Piece request to retrieve (at most) p characters on the screen line n beginning with column m. The top left corner of the screen is row 1, column 1.

### 8.4.3 DDE Message Connections

The Terminal supports DDE requests for data communications through the Message topic. These are implemented with DDE POKE commands.
### Table 8–5: DDE Message Connections

<table>
<thead>
<tr>
<th>Item</th>
<th>Meaning of returned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send</td>
<td>The DDE message value is sent to the host if a connection is active.</td>
</tr>
<tr>
<td>Display</td>
<td>The DDE message value is sent to the “screen” as if it were received from the host.</td>
</tr>
</tbody>
</table>