Using the Windows Terminal

Version 2024.1
2024-05-16

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using the Windows Terminal
InterSystems IRIS Data Platform Version 2024.1 2024-05-16
Copyright © 2024 InterSystems Corporation
All rights reserved.

InterSystems®, HealthShare Care Community®, HealthShare Unified Care Record®, IntegratedML®, InterSystems Caché®, InterSystems Ensemble® InterSystems HealthShare®, InterSystems IRIS®, and TrakCare are registered trademarks of InterSystems Corporation. HealthShare® CMS Solution Pack™ HealthShare® Health Connect Cloud™, InterSystems IRIS for Health™, InterSystems Supply Chain Orchestrator™, and InterSystems TotalView™ For Asset Management are trademarks of InterSystems Corporation. TrakCare is a registered trademark in Australia and the European Union.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

1

Basics of the Terminal Application

The *Terminal application* (available only on Windows) presents the ObjectScript shell in a window that also provides a menu bar with additional options. The primary purpose of the Terminal application is to enable you to execute ObjectScript commands (and see their results), as well as to access various other shells. In the Terminal application, you can also run scripts that use syntax specific to this application. This page provides a basic introduction.

Note: Throughout the documentation, the term *Terminal* is often used to mean just the ObjectScript shell (which has no special menu bar and which is available on all operating systems). Similarly, the phrase *Terminal session* often means a session within the ObjectScript shell.

Where necessary to avoid confusing these concepts, the documentation uses the phrase *Terminal application* to refer to the Windows-only application.

1.1 Starting the Terminal Application

To start the Terminal application, do one of the following:

- To connect to a local database, select the InterSystems Launcher and then select **Terminal**.
- To connect to a remote server, select the InterSystems Launcher and then select **Remote System Access > Terminal**. Then select a server name.

In either case, you may be prompted to log in, or you may instead be logged in under a default username, depending on the security settings for the associated InterSystems IRIS® data platform server. The Terminal application then displays a prompt that (by default) displays the name of the current namespace. For example, if you are in the USER namespace, the default prompt is as follows:

```
USER>
```

To switch to a different namespace, use the *$namespace* variable as in the following example:

Terminal

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

This is the same way that you would switch namespaces within your code; see the $namespace reference page in the *ObjectScript Reference* for more information.

1.2 Overview of Terminal Application Features

The Terminal application consists of a window that provides the ObjectScript shell, as well as a title bar and a menu bar.

First, you can use all the options provided by the ObjectScript shell. You can use it to execute ObjectScript commands (and see their results), as well as to access various other shells. The shell provides an extensive line recall facility as well as several customization options.

In addition to providing this shell, the Terminal application menu bar provides options for copying and pasting text, clearing the screen, performing logging, printing, executing scripts, and customization. The Terminal application also provides a right-click menu with options for copying and pasting.

Note that the title bar indicates the InterSystems IRIS® data platform server to which the Terminal application is connected, which is especially helpful when you have multiple sessions, each to different server. The title bar also indicates the communication mode currently in use:

- The title bar may display **InterSystems IRIS TRM:***pid***(***instancename***)** where:
  - *pid* is the process ID of the InterSystems IRIS process with which the Terminal application is communicating.
  - *instancename* is the InterSystems IRIS instance in which the process is running.

  In this case, the Terminal application is using local, proprietary communication with the InterSystems IRIS server with which it was installed.
- The title bar may display **(***server* **NT — InterSystems IRIS Telnet)** where *server* is the host name of the remote server. In this case, the Terminal application is using the TELNET protocol over TCP/IP to communicate with either a Windows InterSystems IRIS server or with a UNIX® host.

On Windows, 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.

1.3 Copying and Pasting

To copy and paste text in the Terminal application, you can use the right-click menu, the **Edit** menu, or various keyboard shortcuts. On the menus, 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 scrollback buffer). The text becomes visible in the 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.

1.3.1 Keyboard Shortcuts

You can use the following keyboard shortcuts:

Action	Basic Shortcut	Windows Shortcut
Copy	**Ctrl+Ins**	**Ctrl+C**
Paste	**Shift+Ins**	**Ctrl+V**

Action	Basic Shortcut	Windows Shortcut
Copy and paste		**Ctrl+Shift+V**

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 User Settings.

1.3.2 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 execution, 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 application 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.

1.4 Interrupting Execution

To interrupt 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 User Settings.

1.5 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 then repaints the window.
- To reinitialize the screen, select **Edit > Erase** or press **Ctrl+Del**. This option reinitializes the window, erases all session data, and resets the scrollback region to zero.

1.6 Logging the Terminal Application Session

To start logging the current session:

1. Select **File > Logging** or select **Alt+L**.

The Terminal application 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, a dialog box is displayed asking if you want to overwrite it. Choose one of the following:

   - **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 application 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 script. 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 Learning Mode.

1.7 Printing

To print, use the following options of the **File** menu:

- To select a printer and set it up for use with the Terminal application, select **File > Printer Setup**.
- To print the contents of the 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.

1.8 Exiting

To exit the Terminal application, 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 application to exit, closing any open files and stopping any foreground execution.

If this Terminal application was connected to a server at startup, it exits on its own when the communications channel is closed.

If you accessed this Terminal application 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.

1.9 Technical Notes

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

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

1.10 See Also

- Connecting to Remote Hosts
- Using Terminal Scripts
- Script Command Reference
- Customizing the Terminal Application
- Running the Terminal from the Command Line
- Advanced Topics in Terminal Usage

2

Connecting to Remote Hosts

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

Note: If you start the Terminal with either the `/console` or the `/server` control argument, the **Connect** menu is not shown; see Connection Options.

2.1 Connect Menu Options

The **Connect** menu provides the following options.

- 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**.

2.2 Remote Connection 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.

To terminate the session, select **Connect >Disconnect**. Or, to terminate this Terminal, select the Close box in the upper right of the window.

3

Creating and Using Terminal Scripts

This topic discusses how to create and use script files for the Terminal application.

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® data platform control service runs.

3.1 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:

```
ScriptCommand: ScriptArguments
```

Here *ScriptCommand* is one of the Terminal script commands and *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:

```
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 Learning Mode.

3.2 Script Command Summary

The following table gives the list of available script commands, with links to the reference content for more details:

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

For reference information on these commands, see Script Command Reference.

3.3 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>, ..., <P*n*>

This substitutes the *n*-th command line parameter in place of <P*n*>.

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.

Character(s)	Interpretation	Transmitted Sequence
<CR>	Carriage return	<13>
<F10>	F10 key	<27>[21-
<F7>	F7 key	<27>[18-
<DO>	Do key	<27>[29-
<TAB>	Tab key	<9>
<LF>	Line feed	<10>
<ESC>	Escape key	<27>
<DCS>	Device control string introducer	<144>
<ST>	Stop device control string	<156>
<EMU>	Start of extended emulator command	<144>i
<NL>	Newline	<CR><LF>
<CSI>	Control string introducer	<155>

3.4 Sample Script

The following is a sample script:

```
; 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
```

3.5 Starting a Script

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 application to a single choice and thus any script file is invoked after the host connection is made.

3.6 Pausing a Script

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.

3.7 Stopping a Script

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

3.8 See Also

- Script Command Reference
- Learning Mode
- Running the Terminal Application from the Command Line

4

Script Command Reference

This topic provides reference information for the script commands available for the Terminal application, for use when you create Terminal scripts.

Note: 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.

4.1 break

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

```
break
```

4.2 call script

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

```
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.

4.3 case match

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

```
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*.

4.4 closelog

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

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

4.5 connect

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

```
connect
```

4.6 debug

Enables debug mode, which traps invalid script commands, which the Terminal application 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
```

4.7 disconnect

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

```
disconnect
```

4.8 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.

4.9 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.

4.10 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.

4.11 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
```

4.12 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 ...
```

4.13 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.

4.14 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.

4.15 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
```

4.16 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.

4.17 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.

4.18 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>
```

4.19 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.

4.20 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.

4.21 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**.

4.22 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.

4.23 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 application. The Terminal script mechanism sends commands one-after-the-next without regard to the input returning from InterSystems IRIS® data platform 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.

4.24 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:

```
    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>
    exit
```

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

4.25 terminate

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

```
terminate
```

4.26 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.

4.27 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.

4.28 title

Sets the Terminal application 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.

4.29 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 application when in the USER namespace. This means that you can use this command to wait until the Terminal application 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.

5

Customizing the Terminal Application

This topic describes different ways to customize the appearance and behavior of the Terminal application.

5.1 Specifying the Font

To specify the font size, 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 application automatically resizes both screen and font to the largest size available.

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

5.2 Specifying the Colors

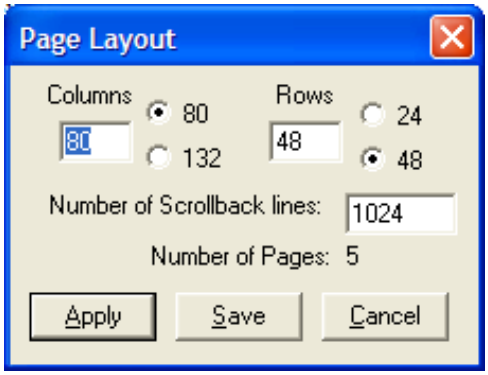
To specify the colors, select **Edit > Color**. This displays a dialog box where you can select the default foreground and background colors for the Terminal application. Then select one of the following:

- **Apply** changes only the current session.
- **Save** does not change the current instance but saves the color information for new sessions.

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.

5.3 Specifying the Window Size

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



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 session.
- **Save** does not change the current instance but saves the window size information for new sessions.

Note: When you change the window size, the Terminal application 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 application selects it.

5.4 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 sessions.

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).

5.5 Specifying 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 application. The settings are as follows:

Setting	Description
Wrap	Automatic wrapping at right hand column
<- Key sends ^H	Set to have <X> key send Ctrl+H rather than Delete
Application Keypad	Enable Application Mode keypad
Force Numeric Pad	Force standard PC keypad

Setting	Description
Disable Special ID	Do not send special Terminal ID
Disable Mouse Reports	Do not sent mouse reports
Enable Fast Paint	Enable fast paint mode
Paste Keeps Linefeed	Set to send line feeds from clipboard (along with carriage return)
Pass XOFF Through	Let remote host handle XOFF/XON.
Windows edit accelerators	Specifies whether the Terminal application enables the common Windows edit shortcuts (**Ctrl+C**, **Ctrl+V**, **Ctrl+Shift+V**), in addition to the basic edit shortcuts (**Ctrl+Insert** and **Shift+Insert**). If these Windows shortcuts are not enabled, the characters are passed in the data stream to the InterSystems IRIS® data platform server process.
Paste burst size (in bytes)	The number of characters to paste in one transmission
Paste pause time (in msec)	The number of milliseconds between successive transmissions of pasted material longer than the burst size

Some systems cannot accept data as fast as the Terminal application 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.

5.6 Specifying the Network Encoding

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

- When the Terminal application 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 application recognizes the change and adapts to it. This enables you to type in mixed languages; the application recognizes it and translates properly into its internal Unicode representation. If you use mixed-language input, select **UTF8** as your network encoding and the **$ZMODE** I/O translation table.
- When the Terminal application 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 application 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.

5.6.1 UTF8 Encoding

When you select the UTF8 option, the Terminal application 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® data platform 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 (\).

5.6.2 Windows Encoding

When you select the Windows option, the Terminal application uses the current Windows input code page to translate I/O between the Terminal application 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.

5.6.3 ISO Encoding

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

Language Region	ISO Standard	Windows Code Page	Network Code Page
Western European	8859-15	1252	28605
Central European	8859-2	1250	28592
Cyrillic	8859-1	1251	28591
Greek	8859-7	1253	28597
Turkish	8859-9	1254	28599
Hebrew	8859-8	1255	28598
Arabic	8859-6	1256	28596
Baltic Rim	8859-4	1257	28594
Korea	iso-2022-kr	949	50225
Japan (JIS)	N/A	932	50220

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 application.

5.6.4 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 application uses the following code pages to translate I/O to and from the server. The Terminal application selects the appropriate EUC code page based on the current Windows input code page. The following mappings are enabled:

Language Region	ISO Standard	Windows Code Page	Network Code Page
Japanese	N/A	932	51932
Simplified Chinese	N/A	936	51936
Korean	N/A	949	51949

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

5.7 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 application window. The difference is apparent only when using multibyte character sets.

5.8 See Also

- Customizing the ObjectScript Shell
- Advanced Topics

6

Running the Terminal Application from the Command Line

You can start the Terminal application from the Windows command line. When you do this, you have options for specifying the server to connect to, the namespace to access, code to run, and the window size and position. You can run routines, and you can also run Terminal scripts.

6.1 Starting the Terminal Application from the Command Line

To start the Terminal application from a Windows command line, use a command with one of the following general forms:

- To connect to the local instance:

  ```
  iristerm /console=ConnectString Arg1 Arg2 ... ArgN ScriptFilePath
  ```

- To connect to a remote instance:

  ```
  iristerm /server=ConnectString Arg1 Arg2 ... ArgN ScriptFilePath
  ```

- To start the application without any connection:

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

  Note: If you start the Terminal application in this way, the menu bar displays an additional menu — the **Connect** menu — which you can use to connect to an instance.

Where:

- *ConnectString* indicates the instance to connect to. See Connection Options.
- *Arg1 ... ArgN* are additional optional arguments, separated by spaces. These arguments specify the server to connect to, and provide other information about the starting environment for the session. You can also specify the name of a routine to run.
- *ScriptFilePath* is the optional pathname of the script file to execute.

If the *PATH* environment variable includes the location of the InterSystems IRIS® data platform binaries, then use the command name `iristerm` or `iristerm.exe`. Otherwise, you must use a full or partial path. For a default installation of InterSystems IRIS, the binaries are in the directory *install-dir*\Bin

6.2 Connection Options

When you start the Terminal application from the Windows command line, you can include the `/console` argument, the `/server` argument, or neither. The following variations are accepted:

/console=cn_iptcp:*HostAddr*

This syntax specifies the target system with which the Terminal application 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]
```

/console=cn_ap:*Instance*[*Namespace*]

This syntax connects to a local instance and switches to the given namespace (if given). For example:

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

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.

/console=cn_ap:*Instance*[*Namespace*]:*Routine*

This syntax connects to a local instance, switches to the given namespace, and executes the given routine. For example:

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

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.

When the routine ends, the session is closed.

/server=*ServerName*

This syntax connects securely to a remote server. For example:

```
iristerm /server=TESTIRIS
```

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.

In order to access the server in this way, make sure of the following:

- 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.

6.3 Additional Arguments

You can also include the following additional arguments:

/size=*Rows*x*Cols*

Specifies the initial size of the Terminal application screen, in terms of rows and columns. Both *Rows* and *Cols* must be unsigned integers. 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 <= *Rows* <= 64
- 20 <= *Cols* <= 132

/pos=(*X*,*Y*)

Specifies the initial origin of the Terminal application 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. You should avoid doing so.

/ppos=(*Xpct*,*Ypct*)

Specifies the initial origin of the Terminal application 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 <= *Xpct* <= 40
- 0 <= *Ypct* <= 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.

/UnbufferedLogging

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.

6.4 Examples

6.4.1 Example: Running a Script in Batch Mode

This example runs a script in batch mode:

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

6.4.2 Example: Running a Routine

This example starts the Terminal application and starts the basic debugging routine **^%STACK** to display information about the current user and the Terminal application process:

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

7

Advanced Topics (Terminal Application)

This topic discusses advanced subjects applicable to the Terminal application. See also Customizing the ObjectScript Shell, which has options specific to the ObjectScript shell.

7.1 Escape Sequences That Affect the Terminal Window

The Terminal application supports the following escape sequences, which affect or provide information about the Terminal window:

Sequence	Effect
`ESC [ 1 t`	Restore window.
`ESC [ 2 t`	Minimize window.
`ESC [ 11 t`	Report window state.
`ESC [ 8;rows;columns t`	Set window size.
`ESC [ 18 t`	Report window size.

The window state is reported in $ZB of the following READ command as:

- normal: `ESC [ 1 t`
- minimized: `ESC [ 2 t`

If rows or columns is 0 in the set command, the current value is not changed. Range of values supported is rows: 10-120, columns: 10-160.

The window size is reported in $ZB of the following READ command as `ESC [ 8;rows;columns t`

One consequence of changing the size is that the scroll-back buffer is cleared by the reset. Another is that for larger row values the font size is decreased to make the window fit on the screen.

Also, the following sequence sets the window title:

```
OSC 2; title ST
```

Where:

- OSC, Operating System Command, is the 7-bit sequence ESC] or the 8-bit character $C(157).

- ST, String Terminator, is the 7-bit sequence ESC \ or the 8-bit character $C(156).

The maximum length of the title is 80 characters.

For example, the following statement changes the title of the Terminal window:

```
write $C(157)_"2;a new title"_$C(156)
```

7.2 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 system manager's directory.

7.3 Learning Mode

In *learning mode*, the Terminal application 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 Session.
2. Press **Alt+Shift+L**.

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

7.4 Disabling the Close Button of the Terminal

If you need to disable the close button (X) of the Terminal application, 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.

7.5 Mapping of the Extended Keyboard

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

Key	Mapped Value
Num Lock	PF1
Keypad-divide	PF2
Keypad-times	PF3
Keypad-minus	PF4
Keypad-plus	Keypad-comma
Shift+Keypad-plus	Keypad-minus
F1, F2, F3, F4	PF1, PF2, PF3, PF4 (respectively)
Shift+F1 ... Shift+F10	F11 ... F20 (respectively)

The keypad cluster of the extended keyboard is mapped as follows:

Key	Mapped Value
Insert	Insert Here
Home	Find
Page Up	Prev Screen
Delete	Remove
End	Select
Page Down	Next Screen

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

7.6 Using the Terminal application with DDE

The Terminal application 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 application screen.
- Message — Used to send data to either the Terminal application screen or the host.

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

7.6.1 DDE Layout Connections

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

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

7.6.2 DDE Screen Connections

The Terminal application 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.

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

Note: The item "Piece" can be POKEd with a string of the form "R*nn*C*mm*L*pp*" to cause the Piece request to retrieve (at most) *pp* characters on the screen line *nn* beginning with column *mm*. The top left corner of the screen is row 1, column 1.

7.6.3 DDE Message Connections

The Terminal application supports DDE requests for data communications through the Message topic. These are implemented with DDE POKE commands.

Item	Meaning of returned value
Send	The DDE message value is sent to the host if a connection is active.
Display	The DDE message value is sent to the "screen" as if it were received from the host.