Using the ObjectScript Shell

Version 2024.1
2024-05-02

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

Using the ObjectScript Shell
InterSystems IRIS Data Platform Version 2024.1 2024-05-02

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

Introduction to the ObjectScript Shell

InterSystems IRIS® data platform provides a command-line interface (the *ObjectScript shell*) which is useful during learning, development, and debugging. You can use it to execute ObjectScript commands (and see their results), as well as to access various other shells. It provides an extensive line recall facility as well as several customization options.

Note: The ObjectScript shell is sometimes called the *Terminal* or the *Terminal prompt*, which is not entirely accurate because there is also a Windows application named Terminal (which provides a way to access this shell). Where necessary, the documentation uses the more specific phrase *Terminal application* to refer to the Windows application.

1.1 Starting the ObjectScript Shell

You can start the ObjectScript shell in multiple ways:

- On any operating system, you can use the **iris terminal** command, starting from an operating system prompt.
- On Windows, you can use the Terminal application, which is a Windows application that presents the ObjectScript shell in a window that also provides a menu bar with additional options. The Terminal application is described separately in more detail.

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 server. The prompt then (by default) displays the name of the current namespace. For example, if you are in the USER namespace, the default prompt is as follows:

Terminal

```
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 General Use

You can enter single-line ObjectScript commands at the prompt. No initial space character is needed, unlike lines of code within routines and classes. For example:

Terminal

```
do ^myroutine
```

Terminal

```
set dirname = "c:\test"
```

Terminal

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

The shell implicitly issues the Use 0 command after each line you enter, so that any output is automatically written to the shell.

There is no multiline mode for entering commands, so the entire command must be typed on a single line. (You can, however, define routines within the shell.)

1.3 Changes in the Prompt

In specific scenarios, the prompt changes to display additional information, to indicate the program stack level or the transaction level. Also see Shells.

1.3.1 The Program Stack Level (Debug Prompt)

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

Terminal

```
USER 5d3>
```

Enter the **Quit** command to exit the debug prompt. Or debug the error; see Command-line Routine Debugging.

1.3.2 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 TL*n*:, 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:

Terminal

```
USER>tstart

TL1:USER>
```

If you exit the ObjectScript shell, that rolls back the transaction.

1.4 Repeating Commands

To repeat a previous command, press the up arrow key repeatedly until the shell displays the desired command. To enter the command, press **Enter**. The ObjectScript shell also supports a more extensive line recall facility, with an option to search previous commands.

1.5 Copying and Pasting

To copy text, select it and press **Ctrl+Ins**. Then to paste it, press **Shift+Ins**.

Additional options are available if you are using the Terminal application.

1.6 Interrupting Execution

To interrupt any foreground execution, press **Ctrl+Shift+c** or **Ctrl+c**.

1.7 Scrolling, Pausing, and Resuming

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

You can also use the keyboard as follows:

Key Combination	Effect
Ctrl+Home	Scroll to the top of the buffer.
Ctrl+End	Scroll down to the cursor.
Ctrl+Page Up	Scroll up by one page.
Ctrl+Page Dn	Scroll down by one page.
Ctrl+Line Up	Scroll up by one line.
Ctrl+Line Dn	Scroll down by one line.
Ctrl+s	Pause the scrolling. While scrolling is paused, the ObjectScript shell accepts commands and processes them, but it does not write the commands or any output to the screen (and thus appears to be unresponsive).
Ctrl+q	Resume the scrolling.

1.8 Exiting

To exit the ObjectScript shell:

1. If you are currently within one of the available (secondary) shells, exit the shell (details vary).
2. Enter `HALT` or `H` (not case-sensitive).

If you had accessed the ObjectScript shell from the **iris terminal** command, the prompt then changes back to the operating system prompt. Or, if you had accessed the Terminal application, the application window then closes.

1.9 See Also

- Available Shells
- Command History and Aliases
- Customizing the ObjectScript shell

2

Available Shells

In the ObjectScript shell, you can start and use other shells that enable you to query data or run code in various ways. You can also start and use operating-system shells.

2.1 Python Shell

To start the Python shell, enter the following command:

Terminal

```
do ##class(%SYS.Python).Shell()
```

The prompt then becomes >>>

Then you can enter Python commands and see their results.

To exit this shell, enter the command `quit()` (not case-sensitive).

For more information, see From the Python Shell.

2.2 SQL Shell

To start the SQL shell, enter the following command:

Terminal

```
Do $SYSTEM.SQL.Shell()
```

The prompt then changes to indicate that you are running within the SQL Shell. Specifically the `[SQL]` prefix is added and a trailing > is added. For example:

Terminal

```
[SQL]USER>>
```

Then you can enter SQL queries and see their results.

Note: The **Ctrl+c** command (normally used for interruption) is disabled within the SQL shell.

To exit this shell, enter `q` or `quit` (not case-sensitive).

For more information, including details on multi-line mode and specifying an SQL dialect, see Using the SQL Shell Interface.

2.3 TSQL Shell

To start the TSQL Shell, enter the following command:

Terminal

```
Do $system.SQL.TSQLShell()
```

The prompt then changes to indicate that you are running within the TSQL Shell. Specifically a trailing > is added.

Then you can enter Transact-SQL queries and see their results. You can use either the MSSQL or the Sybase dialect.

To exit this shell, enter `q` or `quit` (not case-sensitive).

For more information, see Working with TSQL Using SQL.

2.4 MDX Shell

To start the MDX shell, enter the following command:

Terminal

```
Do ##class(%DeepSee.Utils).%Shell()
```

The prompt then becomes >>

Then you can enter MDX queries and see their results.

To exit this shell, enter `q` or `quit` (not case-sensitive).

For more information, see MDX Shell.

2.5 Operating-System Shells

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

Terminal

```
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.

3

Command History and Aliases

The ObjectScript shell supports an extensive line recall facility, with an option to search previous commands, as well as a facility for defining aliases for commands. These options are not an extension to ObjectScript and cannot be used in routines and methods.

3.1 Line Recall Facility

In addition to the up arrow, the ObjectScript shell 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.

To activate line recall, type colon (`:`) followed by one or more characters that unambiguously identify the command.

You can issue a number of special commands that interact with line recall at the command prompt. First, the `:?` displays a quick help:

Terminal

```
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:

Terminal

```
USER>:h[istory]
 1: zn "%SYS"
 2: Do ^SYSLOG
 3: write "Hello"," World!"
 4: write $J
 5: :h

$SYS>:3
write "Hello"," World!"
Hello World!
```

The `:alias` command creates shortcuts for ObjectScript commands commonly typed at the prompt, and `:unalias <name>` removes the definition associated with that name. See Creating Aliases.

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

3.2 Reverse Incremental Search

Reverse incremental search (RIS) is an additional mode of operation of the ObjectScript shell 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 bell rings and nothing else happens. Two backslash characters after the prompt indicate RIS mode:

Terminal

```
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 `W $J` is in the recall buffer, after typing the '$' key, the ObjectScript shell might display this:

Terminal

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

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

Terminal

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

On operating systems 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.

3.3 Aliases

To define a shortcut for issuing a command, enter the **:alias** command followed by the alternative name you want to assign to the command (the alias) and then the command you want to assign to that alias. For the rest of the session, you can issue the command by invoking the alias, preceded by the `:` character.

In the example session which follows, the **:alias** command issued in the first line allows the user to switch to the USER namespace in the second line by simply entering the string `:u`:

Terminal

```
%SYS>:alias u set $namespace="USER"
%SYS>:u
USER>
```

It is also possible to pass arguments to a command using an alias. As part of the alias definition, include numbered host variables prefixed by the `$` character as placeholders for each argument in the command. Then, invoke the alias followed by the values of the arguments in the order indicated by the host variable numbers.

For example, issuing the following command creates a shortcut (`load`) to load a file at a location which will be provided as the first argument (`$1`)with flags specified by the second argument (`$2`):

Terminal

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

To use this alias to load a file found at the path /directory/foobar.xml with the `c` and `k` flags, you can now simply enter the following:

Terminal

```
USER>:load /directory/foobar.xml ck
```

The placeholder variables in a command can also be part of an ObjectScript expression. For example, the following command uses the $SELECT function to set `"ck"` as the default second argument when no second argument is provided:

Terminal

```
USER>:alias load Do $System.OBJ.Load("$1", $S("$2"="":"ck", 1:"$2"))
```

To remove an alias definition, enter the command **:unalias** followed by the alias name. Enter **:alias** with no arguments to see a list of the aliases defined in the current session.

You can also provide a list of alias definitions which the ObjectScript shell will set automatically at the start of every session. Define these aliases (one per line) in a file named .iris_init in your home directory — specifically the home directory for the current user, within the operating system. (For example, on Windows, this is the C:\Users\ subdirectory which corresponds to the current user.)

4

Customizing the ObjectScript Shell

This topic describes ways to customize the ObjectScript shell.

4.1 Defining Routines within the Shell

You can define routines (and then run them), without using an external editor. See the following topics:

- Using the Tab Key Shorthand to Create a Routine
- Create a Routine from the Terminal (this technique involves the ZLOAD, ZINSERT, and ZSAVE commands)

4.2 Modifying the Startup Namespace

When a user starts the ObjectScript shell, the user has access to a particular namespace and can execute code in that namespace. This option is controlled by the **Startup Namespace** option of the user definition. See User Accounts.

4.3 Customizing the Prompt

To customize the default prompt, use the Management Portal to set the configuration option **TerminalPrompt**. To find it, select **System Administration** > **Configuration** > **Additional Settings** > **Startup**. For example, the prompt can include the configuration name of your system.

4.4 ^ZWELCOME Routine

When the ObjectScript shell starts, 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 shell login sequence, if any. The name of the routine implies its intended use, as a custom identification and welcome message to users.

Here is a simple example:

Terminal

```
^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
```

To create the **^ZWELCOME** routine (which must be in the %SYS namespace), you must have administrator privileges and write access to the IRISSYS database.

CAUTION: The **^ZWELCOME** routine runs 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.