

Errata

Sesame v 2.x

Sesame User Guide

Edit Commands | Ditto Current Form (pp. 226)

Remove the sentence "There is no undo". Shift-F7 will undo the form copy. Add to the end of the paragraph "Read-only LEs are not copied."

Appendix 1 – Backup Application Tab (pp. 496)

Remove the sentence: "The files created by Backup Selected Application are not immediately loaded for use as they would be using the similar "Save As" command." Save As does not load the newly saved file.

Utility Programs (pp. 537)

First sentence: "...located in the Tools subdirectory..." should read: "...located in the Utilities\Lantica subdirectory..." Also references on this page to the path C:\Sesame should read C:\Sesame2

Appendix 7 – Sesame Initialization File – sesame.ini

The following ini file entries should be added in the appropriate sections:

Look and Feel Commands (pp. 542)

Box Styles: The box style commands allow you to set the drawing style of parts of the Sesame interface itself. Each of the boxes listed below affects a different kind of screen element. These settings have no effect on user-defined Forms. Valid settings for all box styles are:

NONE
FLAT
UP
DOWN
THIN UP
THIN DOWN
ENGRAVED
EMBOSSSED
BORDER
SHADOW
ROUNDED
ROUNDED SHADOW

Box Style Commands:

BOX STYLE
BUTTON BOX STYLE
DOWN BUTTON BOX STYLE
MENU BOX STYLE
MENUBAR BOX STYLE
INPUT BOX STYLE
OUTPUT BOX STYLE
WINDOW BOX STYLE
SCROLL BOX STYLE
SLIDER BOX STYLE

Example: SLIDER BOX STYLE: FLAT

BORDER COLOR: Sets the border color of many Sesame screen elements to the RGB value following the command. This setting has no effect on user-defined Forms.

Example: BORDER COLOR: 255 0 0

Client Behavior Commands (pp. 545-547)

GLOBAL AUTOCOMplete: Tells Sesame to attempt to autocomplete entries in any text element as you type based on the existing values in that field. Valid settings are ON and OFF.

Example: GLOBAL AUTOCOMplete: OFF

Default: OFF

DIRECTIONAL NAVIGATION: Allows the cursor (arrow) keys to navigate the form in geographic order. For example, with this option turned on Down Arrow is treated as "down" instead of "next". Valid settings are ON and OFF.

Example: DIRECTIONAL NAVIGATION: ON
Default: OFF

JUST IN TIME COMPILATION: Causes form code to compile only when it is about to run. This can eliminate much of the wait associated with opening forms, though GLOBAL CODE will still compile at that time. It may cause some hesitation when using a form for the first time in a session, as the code in various elements is triggered and needs to be compiled. This hesitation will only be present for the first time you trigger a particular event. Valid settings are ON and OFF.

Example: JUST IN TIME COMPILATION: ON
Default: OFF

USE BUTTON PANELS: Tells Sesame to use button panels instead of command trees based on the PANEL BUTTON commands in the ini file. This has the same effect as the ButtonPanel() SBasic command. Valid settings are ON and OFF.

Example: USE BUTTON PANELS: ON
Default: OFF

PANEL BUTTON: If USE BUTTON PANELS is ON, you can use PANEL BUTTON entries to build your button panels. A PANEL BUTTON entry consists of the tree where the button is to appear, its label, and the tree command it is to run when clicked. This matches the ADD_ITEM capability of the ButtonPanelItem() SBasic command. Valid settings for tree are APPLICATION, SEARCH, UPDATE and ADD.

Examples:

PANEL BUTTON: APPLICATION "Search For Customers" "Sample Customers

Application!Forms!Search/Update!Customers!Main Form"

PANEL BUTTON: SEARCH "Close Form" "Search Menu!Exit Search Menu"

PANEL BUTTON: SEARCH "Get Customers" "Search Menu!Search Commands!Retrieve New Results (F10)"

PANEL BUTTON: UPDATE "Save and Exit" "Search Update Menu!Navigation!Save Record and Close Form (Shift-F10)"

PANEL BUTTON: UPDATE "Next Customer" "Search Update Menu!Navigation!Advance Record (F10)"

PANEL BUTTON: UPDATE "Previous Customer" "Search Update Menu!Navigation!Previous Record (F9)"

PANEL BUTTON: UPDATE "New Search" "Search Update Menu!Search (F7)"

HIGHLIGHT ON ENTER: Tells Sesame whether you want existing element contents highlighted/selected whenever you enter an element. Valid settings are ON and OFF.

Example: HIGHLIGHT ON ENTER: ON
Default: OFF

TABLE DISPLAYS IMAGES: Tells Sesame whether you want images to appear when in Table View. Turning on this option may affect Table View performance. Valid settings are ON and OFF.

Example: TABLE DISPLAYS IMAGES: ON
Default: OFF

Server Behavior Commands (pp. 547)

SERVER PREINDEX RELATIONAL: Tells the Sesame server whether to optimize the speed of relational linking using a preindexing scheme. This optimization will only affect relational keys where the Key value contains no search characters. Valid settings are ON and OFF.

Example: SERVER PREINDEX RELATIONAL: ON
Default: ON

SDesigner Commands (pp. 549)

DESIGN PANEL ORDER: Specifies the default order and visibility of the Control Panels in Designer. Each panel has a number. If the number appears in the DESIGN PANEL ORDER command, that panel will appear. The panels are numbered as follows:

1 = Commands

2 = Layout Element Adder

3 = Property Editor

4 = Property Viewer

5 = Advanced Element Selector

6 = Message Log

7 = Change Log/Advanced Undo

Example: To have only the Property Editor, Commands and Property Viewer Control Panels appear - and in that order - use the following ini file entry.

DESIGN PANEL ORDER: 314

Security Commands (pp. 550)

WEB SERVER PASSWORD FILE: The path to the file containing the usernames and passwords to be accepted by the Sesame Web Server. See the *Web Capabilities* section for more details.

SCHEME (Pp. 543)

Additional scheme added.

Softgradient - Like Gradient, but less intense.

Sesame Programming Guide

The Following Functions & Features Have Been Added As Of Version 2.1

Button Menu Style Interface

Sesame 2.1 includes a new style of menu interface that uses buttons in place of command trees. For new installations, this is the default menu style. A complete description of this new interface is included separately. The new SESAME.INI file entries that control the Button Menus are listed below.

MENU STYLE: Controls whether the Command Area displays the new Button Menus or the older Command Trees. Valid settings are BUTTONS and TREE.

Example: MENU STYLE: TREE

Default: BUTTONS

Note to current Sesame Application Users:

The Button Menu interface behaves differently from the Tree interface in terms of how Forms are displayed and how certain selections are hidden in SBasic. If you have a design from a prior Sesame version that depends on certain tree items being hidden, you might want to consider adding MENU STYLE: TREE to the sesame.ini file until you have an opportunity to adjust your application for the Button Menu interface.

When button menus are used you have the option of displaying a list of database applications which can be conveniently opened with a single click. There are two ways of using this feature: a default method which can automatically list files in your Data or Samples folder, or a Custom method where you can specify one or more folders whose applications will be listed. In both cases all DB files will be listed, in alphabetical order.

DEFAULT BUTTON DIRECTORY TYPE: Displays a list of applications when Sesame is first opened. The ONLY valid settings are:

OFF - No application list

CWD - Applications in the Current Working Directory (normally \Sesame2)

DATA - Applications in the Data directory under the Current Working Directory

SAMPLES - Applications in the Data\Samples directory under the Current Working Directory. There is no need to specify Data\Samples.

CUSTOM - Applications in the path(s) specified by CUSTOM BUTTON DIRECTORY PATH ini file settings (See below).

Example: DEFAULT BUTTON DIRECTORY TYPE: DATA

Default: OFF

Notes: The application list will only be shown if the button menu interface is used.

If there are no applications in a particular directory then no group will be shown for that directory.

You cannot specify another directory without using DEFAULT BUTTON DIRECTORY TYPE: CUSTOM.

CUSTOM BUTTON DIRECTORY PATH: If DEFAULT BUTTON DIRECTORY TYPE is set to CUSTOM, Displays a list of applications when Sesame is first opened. This cannot be used without first specifying DEFAULT BUTTON DIRECTORY TYPE: CUSTOM. Each setting is specified as title:path. The list of applications will be grouped with the specified title. You can have up to 10 custom button directory paths. The path specified is relative the Current Working Directory.

Example: CUSTOM BUTTON DIRECTORY PATH: Teachers:Data\School\Faculty

Example: CUSTOM BUTTON DIRECTORY PATH: Students:Data\School\Pupils

Default: No Default

Notes:

The application list will only be shown if the button menu interface is used.

If there are no applications in a particular directory then no group will be shown for that directory.

You can use / or \ as the pathname separator.

DISPLAY RECENT APPLICATIONS: Sets whether a button group is displayed for recently used applications when Sesame is first opened. Valid settings are ON and OFF.
Example: DISPLAY RECENT APPLICATIONS: OFF
Default: ON.

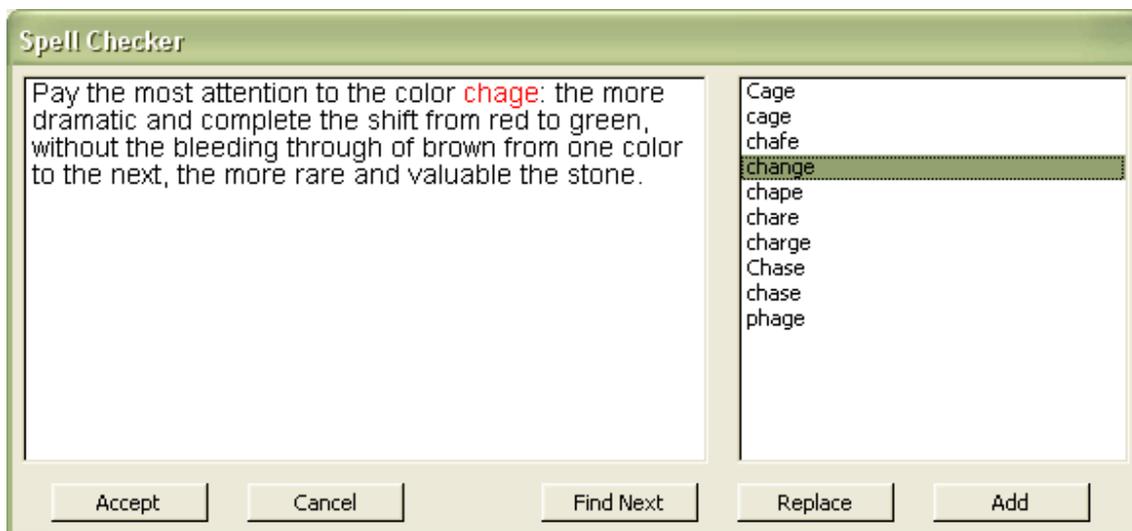
DEFAULT SCROLLBAR WIDTH: Sets the width in pixels of the new "as needed" scrollbars. These scrollbars appear when you enter a scrollable area and disappear when you leave it.
Example: DEFAULT SCROLLBAR WIDTH: 10
Default: 12

Spell Checker

As of version 2.1, Sesame has a Spell Checker that can be used to check spelling in any text field in both an interactive (automatic) and/or manual mode. The manual mode is always available. The Interactive mode is selected by setting an entry in the Sesame.ini file.

Manual Mode

You can manually spell check the words in any text field by selecting that field and clicking Spell Checker in the Edit Commands section of the Button or Tree menus as shown in the following figure.



Spell Checking manually using the Button Menu

Accept: Accepts your changes and returns your corrected value to the spellchecked field.

Cancel: Cancels changes.

Find Next: Highlights the next misspelled word.

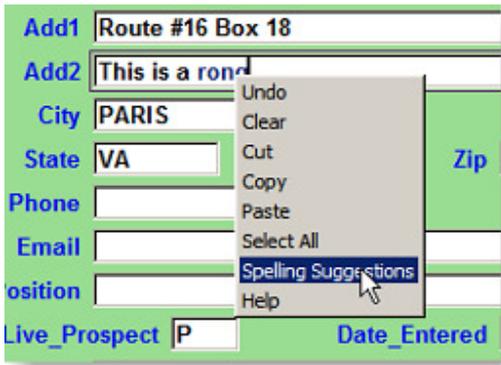
Replace: Replaces the misspelled word with the selected suggestion from the list.

Add: Adds the highlighted word to the dictionary.

You can also type directly in the text box to make corrections.

Interactive Mode

Interactive mode checks spelling as you type. As you complete each word, it will be highlighted if misspelled. While a word is highlighted, you can right-click on it to see suggested spellings or add the word to the dictionary.



Interactive Spell checking as you type

Interactive spell checking is off by default. You can turn interactive spell checking on by adding the following line to the SESAME.INI file:

INTERACTIVE SPELL CHECKING: ON
 Valid settings are ON and OFF.
 Example: INTERACTIVE SPELL CHECKING: ON
 Default: OFF

Dictionaries

Three English dictionary files are supplied with Sesame – US (us_dictionary.txt), British (brit_dictionary.txt), and Canadian (canada_dictionary.txt).

The dictionary.txt file, which appears in your install directory (usually Sesame2), is the active dictionary file used by the Spell Checker. It is supplied as the US dictionary by default. To change the active dictionary, copy the desired dictionary file to dictionary.txt.

The dictionary file used by the Spell Checker can also be controlled by the following sesame.ini file entry:

DICTIONARY PATH: Sets a file to use for the active dictionary.
 Example: DICTIONARY PATH: D:\Words\Canada_dictionary.txt
 Default: dictionary.txt

The dictionary files (dictionary.txt, us_dictionary.txt, brit_dictionary.txt, and canada_dictionary.txt) are derived from files created as part of the "Scowl" dictionary set. The collective work is Copyright 2000-2004 by Kevin Atkinson as well as any of the copyrights mentioned below:

Permission to use, copy, modify, distribute and sell these word lists, the associated scripts, the output created from the scripts, and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation. Kevin Atkinson makes no representations about the suitability of this array for any purpose. It is provided "as is" without express or implied warranty.

Appendix 7 – Sesame Initialization File – sesame.ini

The following ini file entries should be added to the documentation (together with those mentioned above):

APPLICATION MODE: When set to LOCKED, causes Sesame to start up without a menubar, button bar, or tab labels. It also closes the Command Area. If you use this, you must set a startup form, or you will have no choice but to exit without doing anything. Application mode locking is disabled if no application has been specified to open on startup. Valid settings are LOCKED and UNLOCKED.
 Example: APPLICATION MODE: LOCKED
 Default: UNLOCKED

REPORT PAGE BREAK ALWAYS: Causes a CSS page break to be inserted instead of a horizontal line in HTML reports when previewed. If you have reports with forced page breaks and you want to be able to print them directly from your browser preview, set this to ON. Valid settings are ON and OFF.
 Example: REPORT PAGE BREAK ALWAYS: ON
 Default: OFF

The Following Have Been Added As Of Version 2.5

Appendix 1 – Sesame Initialization File – sesame.ini

The following ini file entries should be added to the documentation (together with those mentioned above):

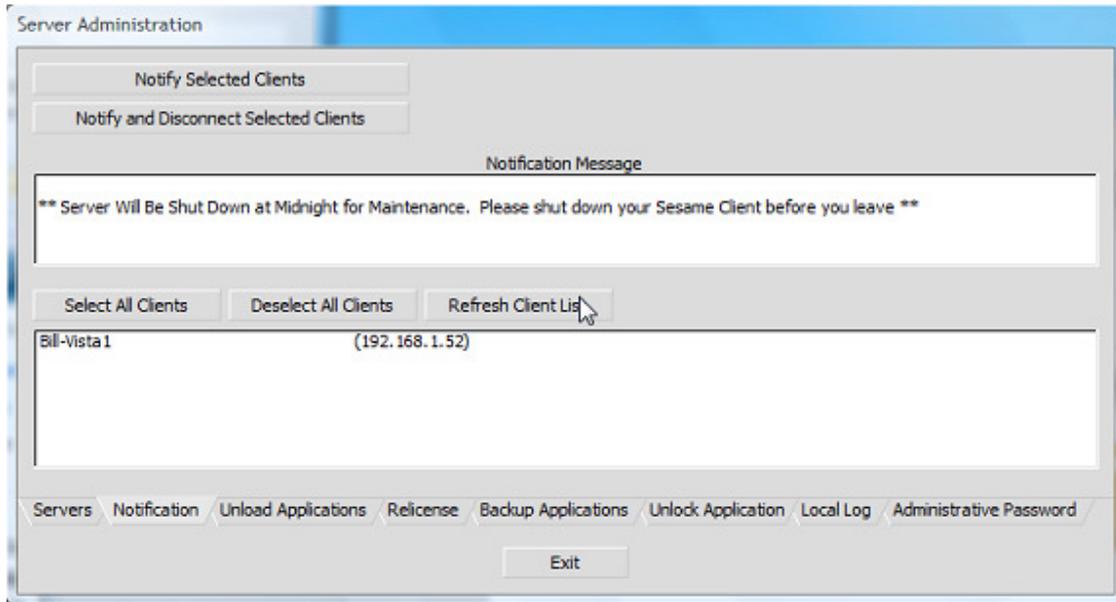
FIELD EDITOR FONT SIZE: When set as FIELD EDITOR FONT SIZE: <n> it defines the size of the font used in the F6 Field Editor Window. Valid settings are between 4 and 42.

Example: FIELD EDITOR FONT SIZE: 16

Default: Uses operating system font settings

Appendix 1 – Network Administration (pp 491)

A new button has been added to the Notification tab of Server Administration that allows you to refresh the list of clients showing in the window. Use this to see the currently connected clients at any point in time.



Appendix 8 – Network Administration (pp 555)

Startup Switches

- A new command line option for server admin command: Unload
[-command <ServerName> <AdminPassword> <UNLOAD> <Application_Title>

EXAMPLE:

Sesame.exe -command MyPC hello UNLOAD "Sample Customers Application"

- Causes server to unload the Customers.db application (referred to by its application TITLE only (no path)).

Warning: This will immediately unload an application that is in use by users or XResultSet commands. Do not do use in those circumstances!

The Following Changes Have Been Made in Version 2.5.1

Sesame 2.5.1 is a maintenance release repairing some annoyances and problems reported in version 2.5. To view these fixes View the Change Log at www.Lantica.com

- A New error message if the client is completely unable to connect to the server.
- Germanic numeric formats for money (comma as decimal point) can be searched using that format.
- A new sesame.ini entry "DEFAULT SEARCH SYNTAX: CASELESS REGEX". RegEx is normally case sensitive. This entry makes RegEx searches case insensitive.
- A change so that apps with a startup form and security place user in Login Box when started

Note: As of this version, Sesame no longer includes the Hoard library and instead uses the memory manager provided by your operating system. Because of this, you may find that the Sesame process takes longer to exit after closing Sesame than with earlier versions.

The Following Changes Have Been Made in Version 2.5.2

Appendix B - Startup Switches

A LOAD command is available for the -command switch, which causes the specified Sesame Server to load the specified database.

```
sesame -command server_name hello LOAD Customers.db
```

The Following Changes Have Been Made in Version 2.5.3

Appendix 7 – Sesame Initialization File – sesame.ini

The following ini file entries should be added to the documentation (together with those mentioned above):

MACROS RETAIN FOCUS: Setting this to ON causes the focus to stay where macros leave it rather than attempting to return focus to the element from which the macro was launched.

Valid settings are ON and OFF

Example: MACROS RETAIN FOCUS: ON

Default: OFF

Sesame Programming Guide

Page 119, State Information Commands

@preview should read @PreviewMode

Page 348, @RedirectProcess(command, feed)

Change the next to last line on the page from '@RedirectProcess("DIR", "")'
to read '@RedirectProcess("cmd /C DIR", "")'

Page 394-395, @SpecCommand(), @omitted.

In all the examples vStr = SpecCommand(SPEC_..) should read vStr = @SpecCommand(SPEC_..)

Page 395, @SpecCommand()

.. SPEC_TYPE_RETRIEVE, "Last Name = m.." should read ..SPEC_TYPE_RETRIEVE, "Last Name = w.." ..

P395, Split(), code example.

The line For i = 1 to 5 should read For n = 1 to 5

P413, @SumListValues.

"The example above will write the number 10375 to the writeln window." should read "The example above will write the number 9475 to the writeln window."

P423, @TreeItemVisibility(path)

The last line of code should read:

```
@RevealTreeItem("Sample Customers Application!Forms ..)
```

P511, Index

#sbasic_include.sbas, 87 should appear on P515 as sbasic_include.sbas, 87

The following functions have been added as of version 2.0.3

@GetSelectionContents()

Type: Operating System

Parameters: None

Returns: string

Returns the contents of the desktop selection buffer (clipboard).

This example prints the contents of the clipboard to the WriteLn window.

```
WriteLn(@GetSelectionContents())
```

@IterateGlobalValues()

Type: State Information

Parameters: None

Returns: string

Returns a semicolon-delimited string listing the names of the current Global Values.

If your application has Global Values as follows:

gvLastInvoice - 10034

gvFindRecord - A041B

gvCompanyName - ABC Manufacturing

The following code will write gvLastInvoice, gvFindRecord, and gvCompanyName to the WriteLn window.

```
WriteLn(@IterateGlobalValues())
```

This example writes out the names and current values of all Global Values.

```
var vList as String
```

```
var vVal as String
```

```
var vLoop as Int
```

```
var vCount as Int
```

```
    vList = @IterateGlobalValues()
```

```
    vCount = @CountStringArray(vList)
```

```

For vLoop = 1 To vCount
    vVal = @AccessStringArray(vList, vLoop)
    If @Len(vVal) > 0
    {
        WriteLn(vVal + ": " + @GlobalValue(vVal))
    }
Next

```

See Also: GlobalValue, @GlobalValue

The following functions have been added as of version 2.0.4

@Mode()

Type: State Information

Parameters: None

Returns: int

@Mode() returns an integer that identifies the current Sesame mode.

```

0 = Add Data Mode
1 = Update Mode
2 = Search Mode (Retrieve Spec)
3 = No Mode (Startup Form)
4 = Dialog Mode (Shown with @FormAsDialog)

```

This code, if added to the On Retrieve Spec Open event of the form, will display a picklist of cities in the CUSTOMERS.db sample database on opening search mode:

```

If @Mode() = 2 Then
{
    City = @XUserSelect(@Filename, "City")
}

```

See Also: @Add, @Update, @IsNew

@SpecCommand(operation, spec_type, command_string, ...)

Type: Spec Management

Parameters: operation as int, spec_type as int, command_string as string, ... as string

Returns: string

@SpecCommand provides a single function that can perform a set of spec commands on any of the Sesame runtime spec types. @SpecCommand takes three required arguments, operation as an integer, spec_type as an integer, and command_string as string. It also accepts a variable number of optional arguments: ... as a string.

operation can be:

```

SPEC_OPERATION_LOAD
SPEC_OPERATION_SAVE
SPEC_OPERATION_RUN
SPEC_OPERATION_VIEW
SPEC_OPERATION_SET
SPEC_OPERATION_CLEAR
SPEC_OPERATION_LIST

```

spec type can be:

```

SPEC_TYPE_RETRIEVE
SPEC_TYPE_SORT
SPEC_TYPE_EXPORT
SPEC_TYPE_IMPORT
SPEC_TYPE_MASS_UPDATE
SPEC_TYPE_COPY
SPEC_TYPE_RESTRICTION
SPEC_TYPE_TABLE
SPEC_TYPE_QREPORT

```

The command_string argument can be repeated to send multiple strings. This function returns the spec as a string array when used with the VIEW operation.

Examples:

```
// Loads an already saved retrieve spec for use
vStr = @SpecCommand(SPEC_OPERATION_LOAD, SPEC_TYPE_RETRIEVE, "MyRetrieveSpec")

// Saves the current retrieve spec as "MyName"
vStr = @SpecCommand(SPEC_OPERATION_SAVE, SPEC_TYPE_RETRIEVE, "MyName")

// Runs the currently loaded retrieve spec
vStr = @SpecCommand(SPEC_OPERATION_RUN, SPEC_TYPE_RETRIEVE, "")

// Returns the current retrieve spec to the variable "vStr"
vStr = @SpecCommand(SPEC_OPERATION_VIEW, SPEC_TYPE_RETRIEVE, "")

// Set the current retrieve spec so that last name begins with "w" and first name begins with "w"
vStr = @SpecCommand(SPEC_OPERATION_SET, SPEC_TYPE_RETRIEVE, "Last Name=w..", "First Name=w..")

// Clears the current retrieve spec
vStr = @SpecCommand(SPEC_OPERATION_CLEAR, SPEC_TYPE_RETRIEVE, "")

// Returns a list of saved retrieve specs
vStr = @SpecCommand(SPEC_OPERATION_LIST, SPEC_TYPE_RETRIEVE, "")
```

@Trap(LE)

Type: Macros and Program Control
Parameters: LE as element reference
Returns: int

Setting a trap with Trap() allows you to keep focus in an element until you release the trap. @Trap allows you to check the state of the trap in the On Element Exit event and act accordingly.

@Trap can return the following states as defined in the sbasic_include.sbas file:

TRAP_NO_TRAP - 0 (Element has no trap set)
TRAP_HAS_TRAP - 1 (Element has a trap which will activate when user enters the element)
TRAP_IS_TRAPPED - 2 (Element trap is currently activated. User is trapped.)

To set and release a trap, you need to put code in several places. The example below traps the user in the element until it is not blank. Once the trap condition is met, it releases the user and performs additional tasks with the validated value.

In GLOBAL CODE

```
#include "sbasic_include.sbas"
```

In the On Element Entry event

```
// Sets the trap
Trap(ThisElement, TRAP_HAS_TRAP)
```

In the On Element Exit event

```
// Check if user is trapped
If @Trap(ThisElement) = TRAP_IS_TRAPPED
{
    // Check trap condition
    If Not @IsBlank(ThisElement)
    {
        // Release the trap
        Trap(ThisElement, TRAP_NO_TRAP)
    }
    Else
    {
        // Show error message
        @MsgBox(@ElementName(ThisElement) + " may not be blank!", "", "")
    }
}
Else
```

```

{
    // User has actually exited the element. Do stuff.
    WriteLn(ThisElement)
}

```

See Also: *Trap*

Trap(LE, state)

Type: Macros and Program Control

Parameters: LE as element reference, state as int

Returns: Nothing

Setting a trap allows you to keep focus in an element until you release the trap. When a trap is set, it activates when focus enters the element. At this point, each time you try to leave the trap element, its On Element Exit programming runs allowing you to test whether to release or reset the trap.

To effectively set a trap, Trap() should be placed and checked in several places. Trap() operates in Form View only, not in Table View. A trap will only be activated if the specified element has On Element Exit programming. If no On Element Exit programming is present, the trap will not activate.

When setting a trap, be careful not to check a condition that cannot be corrected by changing the value in the trap element.

To trap the user in an element named CustomerID:

```

#include "sbasic_include.sbas"
Trap(CustomerID, TRAP_HAS_TRAP)

```

To release the trap set above:

```

#include "sbasic_include.sbas"
Trap(CustomerID, TRAP_NO_TRAP)

```

See @Trap() for a complete usage example.

See Also: *@Trap*

XResultSetOpenForm(rs, form_name)

Type: External Database Application Management

Parameters: rs as int, form_name as string

Returns: Nothing

This command opens the specified form in Update mode with records pre-retrieved.

This function accepts two arguments:

The handle for the result set to use - This must be a result set from a database in the current file. If using @XResultSetSearch, specify @FN as the first argument. You cannot use this command to open forms in other db or dsr files.

The name of the form to open - The specified form must be a form attached to the same database as the one from which the result set records were retrieved.

```

#include "sbasic_include.sbas"

```

```

var vRSHandle as Int
var vNoOfRecords as Int

```

```

vRSHandle = @XResultSetSearch(@FN, "Customers", SEARCH_MODE_AND, SEARCH_SYNTAX_QA,
"!State=PA")
If vRSHandle > -1
{
    vNoOfRecords = @XResultSetTotal(vRSHandle)
    If vNoOfRecords > 0
    {
        XResultSetOpenForm(vRSHandle, "Main Form")
    }
    Else
    {

```

```

        @MsgBox("No records to display.", "", "")
    }
    XResultSetClose(vRSHandle)
}

```

See Also: *XResultSetClose*, *XResultSetCreateNewRecord*, *@XResultSetCurrentForm*, *XResultSetCurrentPosition*, *@XResultSetCurrentPosition*, *XResultSetDeleteRecord*, *@XResultSetForm*, *XResultSetLocked*, *XResultSetRemoveRecord*, *XResultSetReparent*, *@XResultSetSearch*, *XResultSetSort*, *@XResultSetTotal*, *@XResultSetValue*, *XResultSetValue*

The following functions & features have been added as of version 2.0.5

@XResultSetParent(rs)

Type: External Database Application Management

Parameters: rs as int

Returns: int

This function is used to obtain a result set containing a single record - the natural parent of the current record in the result set passed in as the only parameter. This allows you to access the natural parent of a child record directly, even if there is no matching key value. After any operations have been completed on that single record, the parent result set should be closed.

Even though the naturally linked subrecords in the Countries sample application have no matching key values, you can still use @XResultSetParent to get information from the parent record while looking at a subrecord. The example Mass Update below is designed to be run on the Cities records directly. It prints and counts only those City records where the parent Country record is in the Continent of Asia and has an area of larger than .99.

GLOBAL CODE

```

stat sRS as Int
stat sTotal as Int
stat sCount as Int

sRS = @XResultSetForm("")
sTotal = @XResultSetTotal(sRS)
sCount = 0

```

MASS UPDATE

```

var vContinent as String
var vArea as Double
var vPRS as Int
var vCount as Int

If sRS > -1
{
    vPRS = @XResultSetParent(sRS)
    If vPRS > -1
    {
        vCount = @XResultSetTotal(vPRS)
        If vCount = 1
        {
            vContinent = @XResultSetValue(vPRS, "Continent")
            vArea = @ToNumber(@XResultSetValue(vPRS, "Area"))
            If (vContinent = "Asia") And (vArea > .99)
            {
                WriteLn(City)
                sCount = sCount + 1
            }
        }
        Else
        {
            WriteLn("Error: " + @Str(vCount) + " parent records.")
        }
        XResultSetClose(vPRS)
    }
}
Else
{

```

```

        WriteLn("No parent record.")
    }
    If @ResultSetCurrentPosition() = sTotal
    {
        WriteLn(@Str(sCount) + " cities are in the continent of Asia and in a
country with an Area larger than .99.")
        XResultSetClose(sRS)
    }
}

```

@XResultSetRunProgram(rs, global_program, program, test_only)

Type: External Database Application Management

Parameters: rs as int, global_program as string, program as string, test_only as int

Returns: string

This command runs an SBasic program, specified as a string argument, on every record in a result set. It returns as a string any content written using Write or WriteLn.

The programming is sent to the engine to be executed. Because it runs on the engine, the programming must use field names instead of element names. No commands that reference the user interface, forms, or reports are legal in the supplied program. For security, the File I/O commands, Shell commands and Process commands are disabled by default. These can be optionally allowed using the SERVER CODE FILE I/O and SERVER CODE SHELL INI file entries, but you should consider carefully before doing so.

NOTE: Because this method of working with a group of records works directly on the engine, it is much faster than a normal mass update, but it also is able to provide less feedback. You should always test this command using a backup of your data to make sure that is doing what you intend.

Arguments:

rs - Handle to a result set

global_program - Programming that you would type into the GLOBAL CODE area

program - Programming to run on each record in the result set

test_only - Flag indicating whether to actually run the program. 0 runs the program. 1 tests whether the program compiles without running it.

As the program compiles on the engine, the syntax error interface available in the Programming Editor is not available, however, @Error will be set if the program fails to compile. To test your program, set test_only to 1 and check @Error.

For easier assembly of the program to be run into a string, you can write it using single quotes or another unlikely "placeholder" character where double quotes would normally appear. You can then use @Replace to replace the placeholder with double quotes. The example below sets Company to "No company name provided" on any record where Company is blank.

```

#include "sbasic_include.sbas"

var key as int
var pgm as string
var str as string

// Use single quotes to for strings internal to the program string
pgm = "XCompany = 'No company name provided'"

// and replace them with double quotes
pgm = @Replace(pgm, "'", @Chr(34))

key = @ResultSetSearch(@FN, "Customers", SEARCH_MODE_AND, SEARCH_SYNTAX_QA, "!Company==")
if(key > -1)
{
    str = @XResultSetRunProgram(key, "", pgm, 0)
    If @Error
    {
        WriteLn("Program failed to compile.")
    }
}

```

```

    }
    XResultSetClose(key)
}

```

@XResultSetSubSet(rs, field_name)

Type: External Database Application Management

Parameters: rs as int, field_name as string

Returns: int

This function accepts a currently open result set handle and the name of a subrecord field. It returns a handle to a result set representing the naturally linked records that are subrecords of the current parent record in the open result set.

This function accepts two arguments: the handle of the parent result set and the name of the SUBRECORD field in the parent record that defines the natural link.

This allows you to quickly access the natural children of a given parent record without needing to do key-based lookups. The example below can be used in the Countries sample application in a report that has a Value Box bound to Country and an Unbound Value Box named Cities. The code prints the number of Cities subrecords for each parent Countries record on the report.

```

#include "sbasic_include.sbas"

var vPRS as Int
var vCRS as Int
var vCount as Int

    vCount = 0

    // Get current parent record
    vPRS = @XResultSetSearch(@FN, "Countries", SEARCH_MODE_AND, SEARCH_SYNTAX_QA, "!Country=" +
Country)
    If vPRS > -1
    {
        // Get child records
        vCRS = @XResultSetSubSet(vPRS, "Cities Subform")
        If vCRS > -1
        {
            // Set report column to count of cities for this country
            vCount = @XResultSetTotal(vCRS)
            XResultSetClose(vCRS)
        }
        XResultSetClose(vPRS)
    }
    Cities = vCount

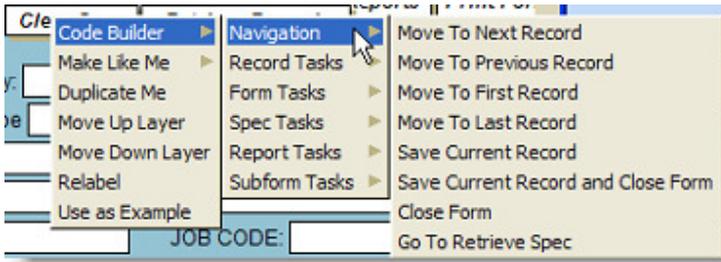
```

Code Builder

The Code Builder creates SBasic code to perform common "one-click" tasks such as print your form, run a report, or sort your records. If you want to see an example of how to do something in SBasic, the Code Builder can often provide you with a prebuilt starting point which you can use as-is or customize to your specific needs.

If an element or layout supports code building, the Code Builder appears on its right-click menu. The Code Builder lists actions for which SBasic code can be automatically created.

When you select an action, Sesame will write the SBasic for that action and assign it to the element. You can see and edit the generated code by looking in the Programming Editor. The Code Builder may ask questions or offer additional options depending on the action selected.

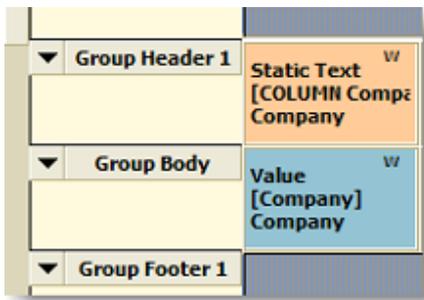


Code Builder Example – Command Button

Note: Currently, Code Builder is supported only for command buttons, but code building support may be expanded to include more elements in future releases.

Visual indicators for visibility and specified width on Report Elements.

Invisible report elements display "I" near the top. Report elements with a specified width (not zero) display "W".



Report Indicating Column Width Set (W)

Switch to Form View button on Table View

When in Table View, a button appears in the upper left corner of the table. Clicking this button switches back to Form View the same way as Shift-F6 does. This button appears both on a main table view and also on a table view subform. Keep in mind that a subform set to start in table view cannot be toggled to form view, so the button will not appear in this case.

	Type	CODE	
68	CJF-ECOM	CIO	Chief Inform
69	CJF-ECOM	DBA	Database A
70	CJF-ECOM	HR-MGT	Human Res
71	CJF-ECOM	POST-SLS	Post Sales
72	CJF-ECOM	PUR	Purchasing

Table View Return Button

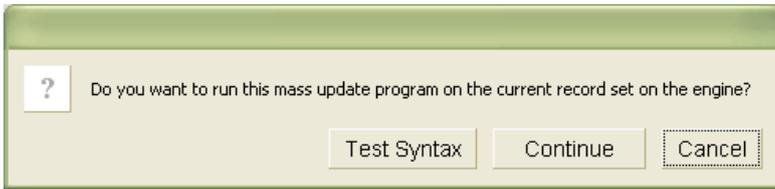
Mass Update Engine

Another option for Mass Update is Mass Update Engine. This option is available under the Results branch of the Command Tree. Mass updates run with Mass Update Engine run on the engine instead of on the client. Mass updates run with this option will run many times faster than an ordinary mass update, but they are also subject to certain limitations.

Differences with Mass Update Engine

- Your forms are not available to engine-side code. Your programming must refer to the underlying field names, not to element names.
- You do not have a GUI to affect. Commands that change colors, pop up lists, etc. are not available and will fail to compile. The exception to this is Write/WriteLn. Any content created during the mass update using Write or WriteLn will be shown in the Slate after the mass update completes.
- For security, the File I/O commands, Shell commands and Process commands are disabled by default for engine-side code. These can be optionally allowed using the SERVER CODE FILE I/O and SERVER CODE SHELL INI file entries, but you should consider carefully before doing so.

- While you write your Mass Update in the Programming Editor as usual, the Test function cannot be used to test your code for syntax errors. When you run Mass Update Engine, there is a Test Syntax option on the confirmation dialog which you can use to test your code for errors.



Mass Update Engine confirmation dialog.

For another option for running engine-side code on a group of records, see the @XResultSetRunProgram command.

Orphan Search - Finds subrecords that do not have a naturally linked parent record. This search command is intended to be used standalone on a form bound to records that are linked as natural subrecords to a parent database. This allows you to easily locate subrecords that do not have a parent record.

Orphan Search respects the criteria in the Retrieve Spec and will only find orphan records that meet the criteria.

Note: Orphan Search is designed to operate on naturally linked subrecords. If you use Orphan Search on a database that has no natural link to a parent database, the results are unpredictable. Orphan Search will not reliably find relational orphans, only natural orphans.

For information on reparenting natural orphans, see the Reparent tree command.

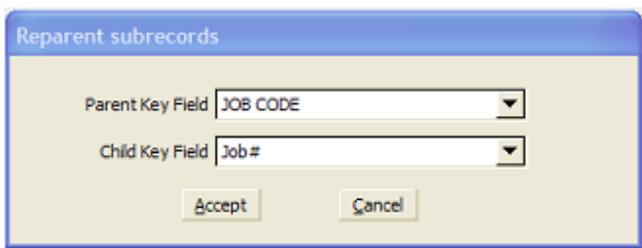
Reparent

The Reparent tree command naturally links each record in your result set to a parent record based on matching values. Since the links are natural, the matching values are only needed until the linking is done. If a record already has a natural parent, it is unlinked from that parent and linked instead to the parent you specify.

If you need to periodically import new natural subrecords for existing parents, this command allows you to import the records standalone, and then quickly link them to their proper parent record. This command is also useful for parenting orphaned subrecords, linking records converted from Q&A, changing from relational to natural linking, and so on.

When you select Reparent, the dialog box shows a list of the parent fields and a list of the child fields. Select one field from each to specify which fields should match. For best results, select fields of the same data type.

When you click Accept, each record in your result set will be naturally linked to the parent record where the value in the parent key field you selected matches the value in the child key field you selected.



Reparent dialog.

Reparent can only be used on a form bound to records that are linked as natural subrecords to a parent database.

For information on finding natural orphans, see the Orphan Search tree command.

Appendix 7 – Sesame Initialization File – sesame.ini

The following ini file entries should be added in the appropriate sections:

Server Behavior Commands

SERVER CODE FILE I/O: Sets whether File I/O commands are allowed to run during engine-side code execution. If these commands are not allowed, a compile error will be thrown when the code attempts to run. This setting affects

@CreateDirectory and all commands listed as type File I/O in the Sesame Programming Guide except for @Insert. Valid settings are ON and OFF.

Note: Consider carefully before turning this option on.

Example: SERVER CODE FILE I/O: ON

Default: OFF

SERVER CODE SHELL: Sets whether shell and process commands are allowed to run during engine-side code execution. If these commands are not allowed, a compile error will be thrown when the code attempts to run. This setting affects the following SBasic commands: @Shell, @ASynchShell, CreateAProcess, @RedirectProcess. Valid settings are ON and OFF.

Note: Consider carefully before turning this option on. Engine-side shell access can be dangerous.

Example: SERVER CODE SHELL: ON

Default: OFF

The following features & changes have been added as of version 2.0.6

@ElementType() (pp. 220)

New defines have been added to sbasic_include.sbas for additional Layout Element (LE) types

LE_TYPE_TABLE2_INPUT - 1021*

LE_TYPE_STATIC_SCROLL_REGION - 1022

LE_TYPE_STATIC_PAGE_MARKER - 1023

LE_TYPE_TEXT_EDITOR - 1024

Example change for addressing editable elements by type:

(n > 999 and n < 1009) or (n = 1024)

Ftp() (pp. 259)

The last argument - source - is the text to upload. Not the name of a source file.

The example should read:

```
Ftp("www.exampleftpserver.com", "mylogin", "mypassword", "online.html", "This is some text.")
```

@PrintAReport() (pp. 329)

New option for mode added to sbasic_include.sbas.

REPORT_MODE_HTML_GENERATE

This option tells Sesame to generate the report file, but not launch it in the browser.

MergeFilePrint() (pp. 298-300)

Fold argument may now also be set to 2. This attempts to fold to end of line.

The following features & changes have been added as of version 2.1

SetCopyBuffer(str)

Type: Operating System

Parameters: str as string

Returns: Nothing

Sets the contents of the desktop selection buffer (clipboard) to str.

This example sets the contents of the clipboard to "John Doe":

```
SetCopyBuffer("John Doe")
```

This example sets the contents of the clipboard to the value in the Company field:

```
SetCopyBuffer(company)
```

See Also: @GetSelectionContents()

Help System

Using F1 or selecting Help from the Help menu now brings up a complete User Guide or Programming Guide in PDF format. These Guides are fully searchable with your PDF viewer. Custom help set on elements remains unchanged.

You can control where Sesame looks for these Guides with the following sesame.ini file entries:

USER GUIDE FILENAME: Sets the location of the Sesame User Guide pdf.

Example: USER GUIDE FILENAME: D:\Docs\Sesame_2_User_Guide.pdf

Default: Sesame_2_User_Guide.pdf

PROGRAMMING GUIDE FILENAME: Sets the location of the Sesame Programming Guide pdf.

Example: PROGRAMMING GUIDE FILENAME: D:\Docs\Sesame_2_Programming_Guide.pdf

Default: Sesame_2_Programming_Guide.pdf

The following New Functions have been added as of version 2.5.3

@LocalCWD()

Type: Operating System

Parameters: none

Returns: string

@LocalCWD returns the current working directory on the client.

```
var vCWD as String

    vCWD = @LocalCWD()
    WriteLn(vCWD)
```

LocalCWD(path)

Type: Operating System

Parameters: path as string

Returns: Nothing

LocalCWD sets the current working directory on the client. After use, it is recommended that you set the current working directory back to the original value.

```
var vCWD as String

    vCWD = @LocalCWD()      // Get current CWD
    WriteLn(vCWD)
    LocalCWD("c:\Windows") // Change CWD
    WriteLn(@LocalCWD())
    LocalCWD(vCWD)         // Reset CWD to original value
    WriteLn(@LocalCWD())
```

@MoneyToWords(val)

Type: Text/String

Parameters: val as string

Returns: string

@MoneyToWords() converts a numeric or currency value to cardinal text like that used for printing checks. So, @MoneyToWords("165.07") would return "one hundred sixty-five and 07 / 100 "

```
var vWords as String

    vWords = @MoneyToWords(Credit_Limit)
    WriteLn(vWords)
```

MouseCursorDefault()

Type: Operating System

Parameters: none

Returns: nothing

This allows an SBasic programmer to indicate that the system is not busy.

```
MouseCursorWait()
```

```
Loiter(3000)
MouseCursorDefault()
```

MouseCursorWait()

Type: Operating System

Parameters: none

Returns: nothing

This allows an SBasic programmer to indicate that the system is busy.

```
MouseCursorWait()
Loiter(3000)
MouseCursorDefault()
```

Q&A Database Translation Guide

Types of Translation Errors

You will encounter two main types of translation errors in the log.

Invalid Data

These errors take the following form:

[Field Name] cannot be retained with value [Illegal Value]

This means that one of your records contains a value that cannot be converted to the specified type. For example "TBA" in a Date field. You will get one of these messages for each invalid value in each record. Because of this, you may see a lot of them. Don't be intimidated by a large number of these error messages. Almost all the errors can be cleared with a simple Mass Update of the field to fix those records where the user entered an invalid value.

Illegal Field Name

These errors take the following form:

Warning: layout element name [Field Name] contains operator characters

Warning: layout element name [Field Name] resolves to a reserved keyword

Warning: layout element name [Field Name] resolves to a number

All these basically mean the same thing. You have an illegal field name. Go back to Q&A, legalize your field names and try the translation again.