Softblox Client Utility Help Index

The Index contains a list of all Help topics available for the Client Utility program. Index items are arrange in alphabetical order within each major category. You can use the scroll bar to see the entries that are not currently visible in the Help window.

For information on how to use Help, press F1 or choose Using Help from the Help menu.

Initiating a Conversation

A conversation with a DDE server can be initiated by pressing the Initiate button in the Session Group box at the bottom of the Client dialog. When the button is pressed, Client will display a popup dialog box containing a list of all DDE server conversations that are available on the system. Choose the desired server conversation by either double clicking the entry in the list box or making the list box entry current and then pressing the OK button as the bottom of the popup
dialog. The entire operation can be cancelled by pressing the Cancel button.

When an entry is chosen, the popup dialog will disappear and the selected entry will appear in the Conversation combo box in the Client dialog window. The Conversation combo box always shows the active conversation that is affected by the other controls in the dialog window. When more than one conversation exists, you can change the current active conversation to another by changing the Conversation combo box to another entry.

Terminating a Conversation

Make the desired conversation the active entry in the Conversation combo box and press the **Terminate** button in the Session Group box. All conversations can be terminated by pressing the **Terminate Conversations and Quit** button.

Whenever the Client utility is terminated, all active conversations with other DDE servers are automatically terminated before the program ends.

Executing a Server Command

Make the desired conversation the active entry in the **Conversation** combo box. Then type the server command in the **Command** edit window. The syntax of the command depends upon the server that is handling the conversation (refer to specific server help information pertaining to Softblox DDE servers). After the command is entered, press the **Execute** button in the Operations group box to forward the command to the server.

The result of the command execution is reflected in the status line at near the right-hand bottom of the Client window. The appearance of **DDE Success** signifies that the command was performed successfully. When **DDE Failure** appears, the command produced an error. The content of **Rc=nnn** provides further information about the reason that the command failed (refer to Return Codes for error code assignments.

Requesting and Poking values

Make the desired conversation active in the **Conversation** combo box. Then type the item name in the **Item** edit window.

To request a value:

Press the **Request** button in the Operations group box. The requested value will appear in the large edit window that has horizontal and vertical scroll bars.

To poke a value:

Enter the data value in the large edit window that has horizontal and vertical scroll bars. Then press the **Poke** button in the Operations group box.

Setting Advisories

Make the desired conversation the active entry in the **Conversation** combo box. Type the item to be advised whenever changed in the **Item** edit window. Press the **Advise** button in the Operations group box to establish the advisory.

An advisory can be cancelled by repeating the preceeding steps but pressing the **Unadvise** button instead of the **Advise** button.

For Setting of Advisories for each specific Softblox Server, refer to the appropriate servers` DDE commands. Specifically refer to handling of standard DDE message WM_DDE_ADVISE.

Softblox Routing Server DDE Commands

System Topic Items

DDE EXECUTE Commands

Only the 'System' topics DDE conversation can perform the DDE_EXECUTE operation to pass one or more commands to the server for processing. When multiple commands are supplied, the command stream is terminated when either all commands are performed successfully or when a command fails. If a command fails, any commands that follow are ignored.

Setting up a profile
Starting the Routing Process
Stopping the Routing Processing
Shutting down the Server

Configuring the Communication Driver
Getting the Status of the Communication Driver
Executing a Protocol-Specific Driver Command

Routing Server System Topic

DDE conversations are passed through the protocol to the remote counterpart. In order to handle external application control, the server supports the DDE 'System' topic conversation so that other programs can send commands to the server.

The 'System' topic provides a mechanism to obtain information about the server and it's current operating state. For responses that contain multiple entries, entries are separated by TAB characters.

Item	Purpose
SysItems	provides a list of all items supported in conversation
SysCommands	returns a list of all supported DDE_EXECUTE commands
Status	returns the current status of the server.
Applications	returns a list of currently supported remote applications handled by the server
ApplTopics	returns a list of currently support applications and topics that are handled by the server.
Profile	returns the name of the server profile that is active.
CommDrv	returns the name of the communications driver that is active on this server.

[profile(profile-filepath)]

loads the specified configuration profile into the server.

[start]

start the currently selected remote communication protocol. This command is rejected if the protocol is already in progress.

[stop]

stop the communication protocol that is currently in progress. This command is rejected if the protocol is not started.

[shutdown]

terminate the server immediately. Any active DDE session is automatically terminated before the server ends.

[amexec(command-text)]

sends a protocol-specific command to the communications driver for processing. The syntax and usage of the command varies by the type of communication driver that is active. This command is rejected if there is no protocol established.

[amstatus]

requests the communication driver to render it's current operating status to the server for subsequent return to the DDE client. The content of the status information varies by communication driver. Some drivers do not return any status.

[amconfig]

invokes the protocol specific configuration dialog. This command is rejected if the protocol is started or there is no protocol established.

Softblox Object Server DDE Commands

DDE EXECUTE Commands

Only the 'System' topics DDE conversation can perform the DDE_EXECUTE operation to pass one or more commands to the server for processing. When multiple commands are supplied, the command stream is terminated when either all commands are performed successfully or when a command fails. If a command fails, any commands that follow are ignored.

Variables

Tables

Object Server - Handling Standard DDE Messages

A DDE conversation can be initiated to either the defined server topic or the 'System' topic. A conversation to the server topic is necessary to manipulate variables or tables. Table manipulation must be performed by issuing DDE_EXECUTE commands to invoke the appropriate table function. The server topic handles the following DDE messages:

WM_DDE_REQ
returns the contents of the variable identified by the item.

WM_DDE_POKE
set the contents of the local variable identified by the item.

WM_DDE_ADVISE
establishes a DDE advisory for the variable identified by the item. When the variable is modified, a WM_DDE_DATA message is produced. The variable must exist prior to sending this message.

WM_DDE_UNADVISE
cancels the DDE advisory for the variable identified by the item.

WM_DDE_EXECUTE
the command that accompanies this message is processed to perform table or variable management services.

WM_DDE_TERMINATE
terminates the conversation and removes any advisories established during the conversation. In Multiple Mode, the local variable pool is released.

Object Server System Topic

The 'System' topic provides a mechanism to obtain information about the server and it's current operating state. For responses that contain multiple entries, entries are separated by TAB characters.

Item	Purpose
SysItems	provides a list of all items supported in conversation
SysCommands	returns a list of all supported DDE_EXECUTE commands
SysTables	returns a list of all tables currently active
Status	returns the current status of the server.

Note: The 'System' topic will accept and process DDE_EXECUTE commands but the proper operation of variable or table oriented commands should be issued in the conversation with the server topic. If the server is operating in Single Mode, the 'System' topic can be used successfully with these functions.

[shutdown]

terminate the server immediately

[vprofile(application-name, private-profile-filepath)]

establish the profile application keyword and filepath for a private profile to be used in variable transfers to/from the variable memory pools and the disk profile pool.

[vdelete(variable-name, pool-name)]

delete a variable from the associated pool (either LOCAL or GLOBAL)

[vreset(variable-name, pool-name)]

reset the contents of a variable in the associated pool (either LOCAL or GLOBAL). This commands differs from vdelete in that the variable remains defined in the pool and notification linkages are not lost.

[vtransfer(variable-name, from-pool, to-pool)]

transfer the contents of a variable from one pool to another. If the variable exists in the destination pool, the destination variable's content is replaced. The from-pool or to-pool specification must be Local, Global, or Profile. Before attempting to transfer information to/from the Profile pool, the vprofile command must have been issued.

[vgfree]

removes all variable from the global pool

[vlfree]

removes all variables from the local pool

[tcreate(table-name, colname1, colname2, ... colnamen)]

create a new table, in an empty state, with the specified column variable names.

[tdestroy(table-name)]

destroy the table and remove it from memory. If the table is being used by other conversations, only the access context for the current DDE conversation is removed and the table unaffected.

[topen(table-name)]

open an access context to the specified table for the current DDE conversation. This command is used when the table is being used by more than one conversation.

[tclose(table-name)]

close the access context to a table for the current DDE conversation. The table is unaffected by this command but the conversation cannot access the table unless it is opened again.

[tlock(table-name)]

increase the lock count associated with the specified table. Attempts to modify the table by any conversation are blocked.

[tunlock(table-name)]

decrease the lock count associated with the specified table. The table remains in a locked state until the lock count is rediced to zero.

[tadd(table-name)]
[tadd(table-name, row-number)]

add a row to the specified table using column-name variables. When the row-number is omitted, the row is added at the end of the table. If the row-number is specified, the row is inserted at the appropriate position in the table.

[tget(table-name)
[tget(table-name, row-number)]
[tget(table-name, row-number, colname1, colname2, ... colnamen)]

get the contents of a row into the column variables. When row-number is omitted or zero, the next sequential row is retrieved. Otherwise, the row corresponding to the row-number is retrieved. Column-item variable names can be specified to map the contents of the row into different variables than those defined for the table. When Column-items are specified, the number of items must match the number of columns in the table.

[tdelete(table-name)]
[tdelete(table-name, row-number)]

delete the current sequential row or specifically indicated row from the table.

[tempty(table-name)]

set the table in an empty state, all rows comprising the table are released from memory.

[trep(table-name)]
[trep(table-name, row-number)]

replace the row contents of a table with the current column-item variable. When row-number is omitted, the next sequential row is replaced. Otherwise, the row corresponding to the row-number is replaced.

[tpoint(table-name, row-number)]

the next sequential row is altered to the specified value which will affect retrieval of subsequent rows on tget commands. Specifying row ZERO will reset retrieval to the beginning of the table.

[trow(table-name)]

returns the current sequential row number in the local variable '@ZROW'.

[tcols(table-name)]

returns the number of columns of the table in the local variable '@ZCOLS'

[trows(table-name)]

returns the number of rows currently in the table to the local variable '@ZMAXROW'

[tload(table-name, filepath)]

creates a table using the table-name and loads the contents of the table file specified by filepath into it. An access context for the DDE conversation is automatically established.

[tsave(table-name, filepath)]

writes the contents of the table to disk. The table is unaffected by the command.

[tcolname(table-name, receiving-variable-name)]

transfers the column item list for the specified table into the specified local pool variable. Columns items are separated by TAB characters.

Object Server - Special Variables

After table commands are successfully performed, the following variables are automatically set in the local variable pool to provide feedback information abount the status of the table.

@zname	contains the table-name
@zrow	contains the row number
@zmaxrow	contains the total number of rows in the table
@zcols	contains the number of columns in the table

Softblox Timed-Initiator Server DDE Commands

DDE EXECUTE Commands

Only the 'System' topics DDE conversation can perform the DDE_EXECUTE operation to pass one or more commands to the server for processing. When multiple commands are supplied, the command stream is terminated when either all commands are performed successfully or when a command fails. If a command fails, any commands that follow are ignored.

Timed-Initiator Server - Handling Standard DDE Messages

A DDE conversation can be initiated to either the defined server topic or the 'System' topic. A conversation to the server topic is necessary to manipulate variables. The server topic handles the following DDE messages:

WM_DDE_REQ
returns the contents of the variable identified by the item.

WM_DDE_POKE
set the contents of the local variable identified by the item.

WM_DDE_ADVISE
establishes a DDE advisory for the action name identified by the item. When the action completes execution, a WM_DDE_DATA message is produced.

WM_DDE_UNADVISE
cancels the DDE advisory for the action name identified by the item.

WM_DDE_EXECUTE
the command that accompanies this message is processed to server oriented services.

WM_DDE_TERMINATE
terminates the conversation and removes any advisories established during the conversation. In Multiple Mode, the local variable pool is released.

Timed-Initiator Server *System* Topic

The 'System' topic provides a mechanism to obtain information about the server and it's current operating state. For responses that contain multiple entries, entries are separated by TAB characters.

Item	Purpose
SysItems	provides a list of all items supported in conversation
SysCommands	returns a list of all supported DDE_EXECUTE commands
Status	returns the current status of the server.
Events	returns a list of all events currently defined
Actions	returns a list of all actions currently defined
ScheduleFile	returns the name of the currently loaded schedule

[start]

start the currently loaded schedule for processing. This command is rejected if the schedule is already in progress or there are no actions defined.

[stop]

stop the schedule that is currently in progress. This command is rejected if the scheduler is not started.

[shutdown]

terminate the server immediately. Any active schedule is automatically halted before the server ends.

[schedule(schedule-filepath)]

loads a schedule into the server for subsequent use. This command is rejected if another schedule is already in progress or the currently loaded schedule has been modified.

[addaction(action-name, action-type, action-file)]

adds a new action to the list of actions available to the current schedule. The action-type is specified as either 'Procedure' or 'Program'. The action-filepath specifies the disk filepath of the associated action processor.

[runaction(action-name)]
[runaction(action-name, interval)]
[runaction(action-name, interval, occurs-count)]

marks the specified action for processing. This command is rejected if the scheduler is not started, the action does not exist, or the action is already marked for processing. When omitted, interval and occurs-count default to 1.

These can be used for User-Controlled Scheduling of predefined Actions.

Softblox Stored-Procedure Server DDE Commands

DDE EXECUTE Commands

Any DDE conversation can perform the DDE_EXECUTE operation to pass one or more commands to the server for processing. When multiple commands are supplied, the command stream is terminated when either all commands are performed successfully or when a command fails. If a command fails, any commands that follow are ignored.

Stored-Procedure Server - Handling Standard DDE Messages

A DDE conversation can be initiated to either the defined server topic or the 'System' topic. A conversation to the server topic is necessary to manipulate variables. The server topic handles the following DDE messages:

WM_DDE_REQ
returns the contents of the variable identified by the item.

WM_DDE_POKE
set the contents of the local variable identified by the item.

WM_DDE_ADVISE
establishes a DDE advisory for the variable identified by the item. When the variable is modified, a WM_DDE_DATA message is produced.

WM_DDE_UNADVISE
cancels the DDE advisory for the variable identified by the item.

WM_DDE_EXECUTE
the command that accompanies this message is processed to perform

WM_DDE_TERMINATE
terminates the conversation and removes any advisories established during the conversation.

Stored-Procedure Server *System* Topic

The 'System' topic provides a mechanism to obtain information about the server and it's current operating state. For responses that contain multiple entries, entries are separated by TAB characters.

Item	Purpose
SysItems	provides a list of all items supported in conversation
SysCommands	returns a list of all supported DDE_EXECUTE commands
Status	returns the current status of the server.
Instances	returns a list of all active procedure instances
Running	returns a list of all executing procedure instances
Applications	returns a list of all server applications present
ApplTopics	returns a list of all server application and topics that are present.
Profile	returns the name of the server profile that is active on this server.

[start]

start the procedure server dispatcher. Any procedure(s) marked to run begin execution immediately. Subsequent procedures that are requested to run will begin automatically. Server configured DDE conversations are permitted. This command is rejected if the dispatcher is lready started.

[stop]

stop the procedure server dispatcher. Any procedure(s) in progress are immediately halted. Server configured DDE conversations are automatically terminated. This command is rejected if the server is not started.

[shutdown]

shutdown the server and terminate. If the dispatcher is started, it is automatically stopped before the server terminates.

[init(instance-name)]

creates an empty procedure instance.

[load(instance-name, procedure-filepath)]

creates a procedure instance and loads a compiled procedure into it.

[compile(instance-name, procedure-source-filepath)]

the source statements from the specified file are compiled into the named procedure instance.

[uncompile(instance-name, output-source-filepath)]

the procedure instance is converted into ASCII source statements and written to the specified file.

[run(instance-name)]

marks a procedure instance for execution. If the dispatcher is started, the procedure executes immediately. Otherwise, it is suspended until the dispatcher is started. This command is rejected if the procedure is already running.

[cancel(instance-name)]

cancels execution of a procedure instance. If the procedure was running, it is halted immediately. If suspended, the request to execute is removed.

[save(instance-name, output-procedure-filepath)]

writes the compiled procedure image to the specified file.

[free(instance-name)]

the procedure instance is removed from the server and associated resources are released.

[profile(configuration-filename)]

load the indicated DDE server configuration into the server. This command is rejected if the procedure dispatcher is running.

Following Notes are provided as Reference for installation of DLL's or Stored-Procedures as DDE servers.

Service Invokation

For DLL functions:

BOOL FAR PASCAL function(HWND hwnd, WORD msg, WORD wParam, LONG lParam);

The values passed to the function are the same values received by the DDE window processor. The function can inspect and use the parameters but should not delete global atoms or release data blocks as these actions are done by the window processor when the function returns. When processing the WM_DDE_REQUEST message, the function should send the WM_DDE_DATA response itself.

For Stored Procedures:

Information pertaining to the service is mapped into variables so that the procedure can take appropriate action.

Variable	**Content**
dde.type	Message type: INITIATE, TERMINATE, REQUEST, POKE, DATA, ADVISE, UNADVISE, EXECUTE, or ACK
dde.application	application name for INITIATE
dde.topic	topic name for INITIATE
dde.item	item name for current message
dde.poke	data received in POKE message
dde.data	data received in DATA message
dde.command	command received in EXECUTE message
dde.ack	POSITIVE or NEGATIVE according to acknowledgement
dde.ack	TIMER if caused by timer message

When a procedure is processing the REQUEST message, it should set dde.data to the data to be sent back to the client by the message post processor.

Procedure processing of each message is atomic. This means that the procedure is executed in it's entirety before returning to the window processor. each DDE conversation has a private variable pool and table access context. Information to be shared among all conversations should be transferred into the global variable pool.

Message Post-Processing

Message Type	Rc	Resulting Action
WM_DDE_INITIATE	==0	accept conversation
WM_DDE_INITIATE	!=0	reject conversation
WM_DDE_TERMINATE	any	terminate conversation
WM_DDE_REQUEST	==0	procedure, send WM_DDE_DATA message
WM_DDE_REQUEST	==0	function, nothing - assumes function responded
WM_DDE_REQUEST	!=0	send negative acknowledgement with Rc
WM_DDE_POKE	==0	send positive acknowledgement
WM_DDE_POKE	!=0	send negative acknowledgement with Rc
WM_DDE_DATA	==0	fAckReq: send positive acknowledgement
WM_DDE_DATA	!=0	fAckReq: send negative acknowledgement
WM_DDE_ADVISE	==0	send positive acknowledgement
WM_DDE_ADVISE	!=0	send negative acknowledgement with Rc
WM_DDE_UNADVISE	==0	send positive acknowledgement
WM_DDE_UNADVISE	!=0	send negative acknowledgement with Rc
WM_DDE_EXECUTE	==0	send positive acknowledgement
WM_DDE_EXECUTE	!=0	send negative acknowledgement with Rc
WM_DDE_ACK	any	release pending data if applicable

Dynamic Link Library functions can set a timer which will cause the Ack function to be entered whenever a WM_TIMER message occurs. Differentiation of whether a timer or ack message has occurred can be determined by checking the msg type.

Softblox Component Return Codes

This section lists all return codes originating from Softblox components and APIs. Return codes are grouped by component and are unique within the entire software suite. There are instances where the meaning of an error is repeated for each component. This was deliberately done in order to classify the error condition to the software component where it occurred.

Code	Meaning or Component
0	Successful

Variable Management

Code	Meaning or Component
16	No memory
17	variable does not exist
18	variable truncated in receiving buffer
19	invalid variable attribute
20	unsupported variable type
21	variable control blocks corrupted
22	Memory lock failed
23	variable management is not initialized
24	catastrophic system error
25	invalid window handle
26	protection violation
27	invalid API request

Table Management

Code	Meaning or Component
32	No memory
33	Memory lock failed
34	catastrophic system error
35	table already exists
36	table does not exist
37	bad table position
38	invalid API request
39	table is empty
40	end of table on sequential retrieve
41	I/O error during load or save operation
42	attempting to load a file that is not a table
43	column name item list truncated in receiving buffer
44	attempt to modify a locked table
45	lock/unlock request invalid
46	table is inuse by other windows
47	table is not open for current window
48	table is already open for current window

Stored-Procedure

Code	Meaning or Component

64	No memory
65	Memory lock failed
66	compiled procedure does not exist on disk
67	attempt to load a file that is not a compiled procedure
68	attempt to execute beyond last procedure operator
69	procedure operation abnormally terminated
70	procedure execution complete
71	invalid procedure operator
72	atom operation failed for current operator
73	heap management operation failed
74	invalid dynamic link library operation
75	invalid DDE communication operation
76	variable management function failed
77	table management function failed
78	invalid goto operation, destination does not exist
79	invalid Clipboard format
80	attempt to store into a literal string
81	error accessing Windows initialization profile
82	compiler: error message returned
83	compiler: error message line exceeds supplied buffer
84	invalid procedure CALL
85	invalid procedure RETURN
86	invalid or illegal table APPLY operator
87	Timeout awaiting DDE acknowledgement
88	attempt to perform DDE operation without a valid session
89	attempt to establish a duplicate DDE conversation
90	DDE communication error
91	Clipboard function failed

Server Specific Errors

Code	**Meaning or Component**
128	No memory
129	Memory lock failed
130	access method: header error
131	access method: receive error
132	access method: transmit error
133	access method: pending request error
134	invalid request
135	invalid command
136	invalid syntax
137	invalid command syntax
138	command execution error
139	command rejected

Messages

Diagnostic errors appear in the form of a message box and will contain one of the following texts:

Maximum conversation count exceeded

There is not enough memory available to save information about the new conversation that has been selected. Terminate any unnecessary conversations that may be active and repeat the Initiate that failed.

Previous DDE operation must be acknowledged first

The previous DDE operation is still being processed by the server application. Another operation cannot be started until the previous is complete. This error will appear on heavily loaded systems or server operations that take a long time process. Repeat the operation again after a few moments. If the error persists, check the server application to ensure that it is still operating properly.

Terms

Advisory
an asyncronous notification when an item changes at a server.

Advise
ask the server to generate an Advisory whenever a specific of item changes.

Unadvise
ask the server to cancel a previous Advise request.

DDE
Dynamic Data Exchange. A protocol for transferring information between Windows applications

Item
the name of an item for a specific piece of information that is exchanged between the client and server applications.

Command
a command that causes the server application to perform a specific service when forwarded by an Execute operation.

Initiate
establish a conversation between the client application and a server application.

Terminate
remove a conversation between a client application and a server application.

Request
ask the server application to render a specific item of information

Poke
send a specific item of information to a server application