Jeff Day, Asymetrix Technical Support

Using DDE in ToolBook

This paper describes ToolBook's Dynamic Data Exchange (DDE) features. Several examples are included that demonstrate how DDE can be used effectively to communicate and exchange data with other Windows applications.

Those aspects of Windows DDE supported by ToolBook are described. What DDE is, why DDE is useful and how DDE works are all discussed. The terminology used to discuss DDE is defined. The DDE syntax used by ToolBook and other Windows applications is explained. ToolBook's run command is described since it is often used in conjunction with DDE.

Using ToolBook as the DDE client is discussed. ToolBook's DDE client commands, getRemote, setRemote and executeRemote, are all described in some detail, including several examples of how each command is used.

The information necessary to use ToolBook as the DDE server is provided. The DDE server command, respondRemote, and messages, remoteGet, remoteSet and remoteExecute, which are used by ToolBook are all described. Examples of using ToolBook as a DDE server are included, along with specific examples of using the command and messages.

Chapter 1

About DDEtc \l 1 "About DDE"§

Windows Dynamic Data Exchange is a communications protocol that defines how two Windows applications can send messages to each other and exchange data. DDE provides Windows applications (including ToolBook) with a mechanism to get data from and give data to other applications and to instruct other applications to execute commands.

Why DDE Is Usefultc \l 1 "Why DDE Is Useful"§

The power of DDE is that it can be used as a means to integrate Windows applications together. Windows applications can use features in other Windows applications by communicating with those applications. DDE is used to communicate between two or more instances of ToolBook, providing a method to link several ToolBook instances together into a single integrated application.

Most computer users are not involved in the process of organizing, programming and presenting information—they simply need to browse through and edit an existing base of information. Through DDE, the complexities of various Windows applications can be hidden behind a simple and graphical ToolBook front end.

For example, ToolBook could be used as a front end to an application like Microsoft Excel. Many people use Excel for routine tasks and seldom need to access every aspect of Excel's complex user interface. However, Excel provides many valuable arithmetic and database tools not available in ToolBook. Authors can build easy-to-use applications in ToolBook while using Excel for calculations and data storage. ToolBook's DDE commands could get data from and give data to Excel. Authors can also instruct Excel to execute macro commands. All of this complexity can be hidden from the user through ToolBook's easy-to-use graphical user interface.

Another example of a use for DDE is a stock price monitoring application. An author could set up a communications program like Future Soft DynaComm to monitor an online stock quote service. When the value of a particular stock changes, a DynaComm script could use

DDE to automatically inform ToolBook of the new value. This change is then reflected in an animated ToolBook chart.

How DDE Workstc \l 1 "How DDE Works"§

A client application starts a DDE conversation by broadcasting a message that it wants to do so to every Windows application that is currently loaded. The client's request includes the server name of the application it wants to talk to. The server application, if available, will acknowledge the request with a message of its own.

At this point the conversation has been started and the client can provide the server with or ask for data, and instruct the server to execute a command.

Either application ends the DDE conversation by sending a message informing the other that the conversation should be terminated. Most applications are polite enough not to end a DDE conversation until it is clear that doing so is OK with the other application.

DDE Terminologytc \l 1 "DDE Terminology"§

In this section, frequently used DDE terms are defined, and server names and server topics are explained in detail. The server names and server topics used by several Windows applications are provided.

Definitionstc \l 1 "Definitions"§

You must be familiar with the following terms to use and understand DDE.

DDE Definitions

Term	***Definition***
Client	The application that initiates, or starts, a DDE conversation.
Server	The application being talked to.
Server name	The name used to identify a server application in a conversation.
Server topic	The specific document, book or other topic which is the focus of the DDE conversation.
Conversation	An exchange of DDE messages between a client and server application. A DDE conversation is identified by its application and topic.
DDE request	An attempt by a client to get or set data or execute a command in the server application.

Server Namestc \l 1 "Server Names"§

Every application that supports DDE has a server name. A client application specifies this name to identify which server application it wants to communicate with. Server applications know to respond to initiation requests that include their server name. Note that server names are not case sensitive in OpenScript. Both TOOLBOOK and toolBook are evaluated as the same name. The following table shows the server names of several popular Windows applications. That application's documentation should tell you the correct server name. (The server name is not necessarily the same as the .EXE file, but if you cannot find the server name in the documentation, the name of the .EXE file is a good thing to try.)

Common Server Names

Application	***Server Name***
Asymetrix ToolBook	toolbook
Microsoft Excel	excel
Microsoft Word for Windows	winword
DCA CrossTalk	xtalk
Future Soft DynaComm	dynacomm
Blyth Omnis 5	omnis5
Precision Software SuperBase 4	sb4w
Samna Ami Professional	amipro
Owl GuideBook	guidebook

Server Topicstc \l 1 "Server Topics"§

Different server applications recognize different values as topics. Most applications recognize a special topic name, system, which is a generic topic used when a DDE message or command is not specific to any given document or other topic. The table below describes valid topics for different DDE server applications. The server application's documentation should tell you which topics it recognizes.

Valid DDE Topics

Application	***Valid server topics***
Asymetrix ToolBook	currently open books, *system*
Microsoft Excel	currently open files, *system*
Microsoft Word	currently open documents, *system*
CrossTalk	*system*
DynaComm	Dynacomm server script name, *null*, *system*
Omnis 5	Omnis 5 application name (without .DAP extension)
SuperBase 4	currently open file (.SBF extension assumed)
Ami Professional	table identifiers, *system*
GuideBook	*control*

The ToolBook Run Commandtc \l 1 "The ToolBook Run Command"§

The ToolBook run command is not a DDE command, but it is frequently used in conjunction with ToolBook's DDE commands. An understanding of how to use run is helpful in working with DDE. This section describes run , and several examples are provided showing how the command can be used.

Before an application can participate in a DDE conversation, it must be running. To start an application from inside an OpenScript script, you can use run. The syntax of run is:

```
run <file name> [minimized]
```

run works in the same way as the Run menu items on the File menus of the Program File and File Manager programs. Note that run is not a DDE command, although it is frequently used in conjunction with DDE commands.

<*file name*> should be the name of an application, followed by any of that application's startup parameters. <*file name*> may also be the name of a data file whose extension is listed in the [Extensions] section of the WIN.INI file.

If no extension is specified in the <*file name*> parameter, ToolBook attempts to load another instance of ToolBook for a book by the name specified in <*file name*>.

If you specify an executable file in <*file name*>, ToolBook will only be able to run the program if it is located in the current working directory or a directory specified in your PATH environment variable. If you specify a data file in <*file name*>, ToolBook will only be able to launch the appropriate application if the data file's extension is listed in the [Extensions] section of WIN.INI and if the file is located in the current working directory. You may also specify the complete path name of the data file in <*file name*>.

If the minimized parameter is included, the application will launch as an icon.

Examples

The following examples demonstrate how the run command can be used. The GetRemote section has an example that shows how run can be used in conjunction with DDE commands.

```
--This button script will run an instance of ToolBook for the book "budget.tbk".
to handle buttonUp
    run "budget.tbk"
end

--This script will do exactly the same thing as the previous script
```

```
to handle buttonUp
     run "budget"
end

--This button script will start Microsoft Excel and load the worksheet "budget.xls"
to handle buttonUp
     run "excel.exe budget.xls"
end

--This script will do exactly the same thing as the previous script.  Note that
--there must be a .XLS entry in the extensions section of WIN.INI.
to handle buttonUp
     run "budget.xls"
end
```

ToolBook as the DDE Clienttc \l 1 "ToolBook as the DDE Client"§

There are three OpenScript DDE commands, summarized in the following table, which comprise ToolBook's client side DDE support.

ToolBook DDE Commands

Command	***Action***
getRemote	Gets data from a DDE server application.
setRemote	Sets the value of an item in a DDE server application.
executeRemote	Instruct a DDE server application to execute a command.

Each of these three commands completes a DDE transaction each time it is executed. This means that a DDE conversation is initiated and terminated with each command, and a DDE channel is open for only as long as it takes to complete the command.

Note: ToolBook 1.0 does not support Microsoft's DDE advise protocol..

This section explains how ToolBook can be used as the DDE client. ToolBook's DDE client commands, getRemote, setRemote and executeRemote, are all described in some detail, including several examples of how each command is used.

Chapter 1

ToolBook DDE Syntaxtc \l 1 "ToolBook DDE Syntax"§

The syntax used by ToolBook's DDE commands is:

```
getRemote <item> [application <server name>] [topic <server topic>]
setRemote <item> to <value> [application <server name>] [topic <server topic>]
executeRemote <command> [application <server name>] [topic <server topic>]
```

application <*server name*> is mandatory in ToolBook for all of these DDE commands, except in the special case where the server application is ToolBook. All parameters may be any valid OpenScript expression. If you do not specify a server, then ToolBook assumes that you want to talk to another instance of ToolBook.

topic <*server topic*> is used to identify which server topic you want to converse with. If you do not specify a topic, then ToolBook attempts to initiate the conversation without specifying a topic. Many server applications require that a topic be specified. If you omit topic when ToolBook is the server, the last instance that was loaded becomes the topic.

sysErrortc \l 1 "sysError"§

Immediately after ToolBook executes a client DDE command, the value of sysError is set to a list of nine items. The first item indicates the status of the DDE command, as described in the table below. The last eight items of sysError describe the LOBYTE of the WM_DDE_ACK message which may have meaning to the server application. (They do not have any meaning to ToolBook.) For information on the meaning of these items, see the DDE documentation of the server application.

After executing a DDE command, immediately check sysError to determine whether that command was successful. If the first item of sysError contains something other than OK, then the DDE command was not successful, and you should either attempt it again or provide some other error handling response.

sysError Values for DDE Commands

First item of sysError	*Meaning*
OK	The application responded to the request.
Failed: Busy	The application responded but is busy.
Failed: Denied	The application responded but could not or would not satisfy the request; the syntax of the item or command was invalid.
Failed: Memory Error	ToolBook did have enough global or local memory to complete the command.
Failed: Interrupted	The application responded but the conversation was terminated before the request was satisfied.
Failed: No Server	No application responded to the request; the application or topic was not recognized; the application may not be running.

getRemotetc \l 1 "getRemote"§

Use getRemote to request data from a server application. The syntax is:

```
getRemote <item> [application <server name>] [topic <server topic>]
```

<*item*> specifies the data you would like the server application to provide. This parameter can be any valid OpenScript expression that evaluates to a string.

Different applications recognize different values as valid item specifiers. The following table summarizes some of the valid item specifiers for various Windows applications that support DDE.

Valid DDE Item Specifiers

Application	***Valid Item Specifier***
Asymetrix ToolBook	Any object or system property. For example: text of field, idNumber of page, sysWindowHandle, position of mainWindow, user-defined properties, etc.
Microsoft Excel ("R1C1:R10C2"),	Cell ranges in row-column format or cell range names
Microsoft Word	Bookmark
CrossTalk	System variable
DynaComm	Item specified in *When Poke* structure
Omnis 5	Field name
SuperBase 4	List of field names
Ami Professional	Table identifier

Note: ToolBook does not know what constitutes a valid item to server applications–it simply passes <*item*> to server applications. If the server understands the item string, it provides ToolBook with the requested textual data. If the server is unable to interpret the item string, it returns an error to ToolBook, such as "Failed: Denied".

ToolBook places the value returned by the server application into the special variable It, and sets sysError according to the success or failure of the getRemote command.

Examples

The following examples demonstrate how the getRemote command can be used in a variety of situations.

```
--This button script  updates a summary field in the client ToolBook instance by
--requesting data from a ToolBook server instance.  Remember that server
--applications and topics must be loaded before DDE can be used successfully.
--Note the double-quotes in the item string, which are used to indicate quotes
--within a string.
to handle buttonUp
    getRemote "text of field ""Regional Sales""" \
        application toolbook topic "western.tbk"
    put it into textline 1 of text of field "summary"
end

--This example is the same as the previous example, but Excel is used as the
--server application.  The summary data for both regions is stored in a named
--range of Cells, Regional_Totals.  Excel returns data in the Windows CF_TEXT
--format, that  is, tab-delimited text.
to handle buttonUp
    getRemote "Regional_Totals" application excel topic "regions.xls"
    put it into text of field "summary"
end

--This button script  checks to see if the DayBook help book is already running.
--If it is then it  does nothing; otherwise the book is started using ToolBook's run
--command.
to handle buttonUp
    getRemote "sysWindowHandle" application toolbook topic "dbhelp.tbk"
    if "OK" is not in sysError
        run "dbhelp.tbk"
    end
end

--This button script  will alternately show and hide the mainWindow of another
--ToolBook instance.  Note that  since the server application is ToolBook, an
--application and topic are not  specified in any of the DDE commands.  (If more
--than one other instance of ToolBook were loaded, only the newest  instance
--would be affected by this example.)
to handle buttonUp
```

```
    getRemote "visible of mainWindow"
    if it is true
        executeRemote "hide mainWindow"
    else
        executeRemote "show mainWindow"
    end
end
```

```
--This button script will get the current text  of a Winword bookmark and put
--it into the text  of a field.
to handle buttonUp
    getRemote "Addressee" application winword topic "formltr.doc"
    put it into text of field "Addressee"
    --Remember, the field "Addressee" is not the same thing as the Winword
    --bookmark, "Addressee".
end
```

```
--This page script  gets the values stored in three SuperBase 4 fields and
--update ToolBook each time the page containing this script is displayed.  This is
--an interesting example because it sets the color of the appropriate county on a
--state map in ToolBook based on the County field in SuperBase.
to handle enterPage
    system oldCounty
    set sysLockScreen to true
    getRemote "company" application sb4w topic "company.sbf"
    put it into text of field "Company"
    getRemote "address" application sb4w topic "company.sbf"
    put it into text of field "Address"
    --reset old County object to white
    set fillcolor of irregularPolygon oldCounty to white
    getRemote "county" application sb4w topic "company.sbf"
    set fillcolor of irregularPolygon it to blue
    set oldCounty to it
end
```

setRemotetc \l 1 "setRemote"§

Use setRemote to set the value of an item in a server application. The setRemote command's syntax is much like that of getRemote, the primary difference being that you specify a value that the specified item should be changed to. The syntax of the setRemote command is:

```
setRemote <item> to <value> [application <server name>] [topic <server topic>]
```

<*item*> is any valid OpenScript expression. The item string must be recognisable by the server application. Valid items for commonly used applications are summarized in the "Valid DDE Item Specifiers" table above.

<*value*> is any valid OpenScript expression which is meaningful from the server's point of view. For example, setting a numeric field in SuperBase

4 to "George" would not work, while setting it to 1500 would work.

ToolBook sets sysError according to the success or failure of the setRemote command.

Examples

The following examples demonstrate how the setRemote command can be used in a variety of situations.

```
--This script  will force another instance of ToolBook to switch to author level
--even if all of  its menus (or specifically the author menuitem) have been
--removed.
to handle buttonUp
    setRemote "sysLevel" to "author" application toolbook topic "hidemenu.tbk"
end

--This example sets the text of field "foo" in another ToolBook instance.
to handle buttonUp
    setRemote "text of field foo" to "Net Profits"
end


--This script sets a range of Excel cells equal to the text of a field.
to handle buttonUp
    setRemote "R2C3:R11:C3" to text of field "test data" \
        application Excel topic "testanls.xls"
end

--This script sets the text of a bookmark in a Winword document whenever
--text changes in the ToolBook field containing this handler.
to handle leaveField
    setRemote "UserName" to my text application winword topic "security.doc"
end

--In this script, ToolBook is acting as a DDE "middle man" between
--WinWord and Excel.  The script would be used in a book used to control
--data exchange between the two applications.
```

```
to handle buttonUp
     getRemote "StockPrices" application excel topic "prtfolio.xls"
     put CRLF & text of field "old prices" after it
     --In the next statement, "prices" is a winword bookmark
     setRemote "prices" to it application winword topic "annlrept.doc"
end
```

executeRemotetc \l 1 "executeRemote"§

One of the most powerful ToolBook DDE commands is executeRemote. This command instructs a server application to execute a particular command or set of commands. The syntax for this command is:

```
executeRemote <command> [application <server name>] [topic <server topic>]
```

<*command*> is any valid OpenScript expression that the server can evaluate as a command or set of commands.

ToolBook is not able to determine if <*command*> is valid. The command string is passed to the server, and the server executes it if it is able to do so. ToolBook sets sysError according to the success of the executeRemote command.

Note: You should understand the command language of the server application before attempting to use ToolBook's executeRemote command.

The following table contains a summary of valid command types for the most commonly used Windows applications that support DDE. You should refer to the server application documentation before attempting to execute a command remotely from ToolBook. Most failed executeRemote commands are caused by invalid server application command syntax.

Valid DDE Commands

Application	***Valid Commands***
Asymetrix ToolBook	Any OpenScript command
Microsoft Excel	Any Excel Macro command enclosed in square brackets
Microsoft Word	Any Word BASIC command enclosed in square brackets
CrossTalk	Any CrossTalk DDE command enclosed in square brackets
DynaComm	All DynaComm scripts must be parsed by the currently active *When Execute* structure. This is manually programmed.
Omnis 5	Any Omnis 5 command enclosed in square brackets
SuperBase 4	Any valid DML command
Ami Professional	Any Ami Pro macro command
GuideBook	Any GuideBook command enclosed in square brackets

multiple commands separated by semi-colons

multiple commands in separate brackets

multiple commands separated by colons

Examples

The following examples demonstrate a variety of setRemote uses.

```
--This example script forces another ToolBook instance to shut down.
to handle buttonUp
```

```
    --set sysSuspendMessages to true to prevent application from
    --intercepting shut down attempt.
    executeRemote "set sysSuspendMessages to true;send exit to system"
end

--This example will set the value of a ToolBook system variable in the server
--instance to a user-defined page property in the client ToolBook instance.
to handle buttonUp
    executeRemote "system currentFile;set currentFile to " & fileName of this page
end

--This example starts Excel with a worksheet file and then sizes and moves the
--Excel window with Excel macro commands.  Note that the square brackets are
--required by Excel.
to handle enterBook
    run "excel.exe budget.xls"
    executeRemote \
        "[app.restore()][app.size(481,190)][app.move(1,172)]" \
            application excel topic system
end

--This handler instructs Winword to open a file called "DDESUMM.DOC" with the
--WordBASIC macro command, FileOpen.
to handle buttonUp
    executeRemote "[FileOpen ""DDESUMM"", 0]" application winword topic system
end
```

ToolBook as the DDE Servertc \l 1 "ToolBook as the DDE Server"§

Several built-in messages are available in OpenScript that allow an author to control how ToolBook behaves when acting as the server in a DDE conversation. The messages described in the following table are sent to the current page of a book when a client application attempts to get or set data or execute a command.

ToolBook DDE Messages

Message	***Event***
remoteGet message	A client application attempts to REQUEST data from ToolBook. A WM_DDE_REQUEST has been received.
remoteSet	A client application attempts to POKE data into ToolBook. A WM_DDE_POKE message has been received.
remoteCommand	A client application attempts to EXECUTE a ToolBook command. A WM_DDE_EXECUTE message has been received.

As with all ToolBook messages, these server notification DDE messages are passed up the ToolBook object hierarchy if there are no handlers for them on the current page. It is common practice to place handlers for these messages in the book script so that all DDE messages are trapped, regardless of the current page or background.

In most cases, you will not need to write handlers for these server notification messages. By default, ToolBook attempts to respond in a manner consistent with the DDE client's request. It is useful to trap (i.e. write handlers for) these messages when you want to change the way ToolBook responds to a DDE request. Further details about these messages are given below.

Note: ToolBook 1.0 does not support Microsoft's DDE advise protocol. No message is sent if a client application attempts to initiate a DDE advise link.

respondRemotetc \l 1 "respondRemote"§

When an author traps one of ToolBook's DDE messages, the author is responsible for completing the DDE conversation by either forwarding that message to the ToolBook system or by using the respondRemote command. This command is used to indicate the success or failure of the client's DDE request. The syntax for this command is:

```
respondRemote <response>
```

<*response*> is a list of nine items, in which the first item may be any value described in the table below. The last eight parameters, which are optional boolean values (i.e. zero/non-zero), comprise the LOBYTE of the WM_DDE_ACK message. These eight items are ignored by ToolBook, but may be meaningful to a server application.

Valid DDE Responses

Response	***Meaning***
OK	The DDE request was completed successfully
Failed: Busy	ToolBook was too busy to satisfy the request
Failed: Denied	ToolBook refused the request

Examples

The following examples demonstrate some useful techniques for a ToolBook application acting as a DDE server.

```
--The following set of three book script handlers have the effect of "turning off" this
--book as a DDE server.
to handle remoteGet
     respondRemote "Failed: Denied"
```

```
end

to handle remoteSet
    respondRemote "Failed: Denied"
end

to handle remoteExecute
    respondRemote "Failed: Denied"
end

--This handler prevents ToolBook from accepting any additional DDE requests until
--the current remoteGet message is handled.  This is useful if the client application is
--sending DDE requests faster than ToolBook is able to handle them.
to handle remoteGet
    --This line prevents all messages (including the DDE messages) from being sent
    --until the top-most handler (this handler) has completed execution.
    set sysSuspendMessages to true
    forward  --forward the remoteGet message to the system for handling
end
```

remoteGettc \l 1 "remoteGet"§

The remoteGet message is sent to the current page of the ToolBook server instance when a client application requests data. The syntax is:

```
remoteGet <item>
```

<item> is the item of data that the DDE client is requesting. ToolBook's default response to this message is to execute the statement *set it to evaluate(<item>)*, and then return *It* to the client application.

Examples

The following example demonstrates how the remoteGet message can be used.

```
--This example looks for a special item, "Total Sales", and provides
--the requested data accordingly.  If the requested item is not "Total Sales",
--then the forward command allows ToolBook to handle all other requests
--for data.
to handle remoteGet data
    if data is "Total Sales"
        set it to text of field "Units Sold" * text of field "Price"
        respondRemote "OK"
    else
        forward
    end
end
```

remoteSettc \l 1 "remoteSet"§

The remoteSet message is sent to the current page of the ToolBook server instance when a client application attempts to set the value of an item in ToolBook. The syntax is:

```
remoteSet <item>, <value>
```

<*item*> is the data item which the client application is attempting to set to the value specified in <*value*>. ToolBook's default response to this message is to execute the statement *set <item> to <value>*.

Examples

The following example demonstrates how the *remoteSet* message can be used.

```
--This script demonstrates how the remoteSet handler can be used to simplify
--the syntax required in the DDE client application.  This handler will set the
--text of a field specified, but it will refuse all other remoteSet requests.
to handle remoteSet fieldName, newValue
    --fieldName and newValue are variables whose values are equal to
    --the <item> and <value> parameters, respectively.
    set sysSuspend to false
    get position of field fieldName
    --The previous statement is a dummy reference to the field specified by
    --fieldName.  If the specified field exists, then sysError is set to
    --"OK"; otherwise it is some error message.
    if sysError is OK
        --The field exists, so set its text to the specified value
        set text of field fieldName to newValue
        respondRemote "OK"
    else
        --No field by the name contained in fieldName exists, so deny the request.
        respondRemote "Failed: Denied"
    end
    set sysSuspend to true
end
```

remoteCommandtc \l 1 "remoteCommand"§

The remoteCommand message is sent to the current page of the ToolBook server instance when a client application requests that ToolBook execute the command. The syntax is:

```
remoteCommand <command>
```

<*command*> contains the command(s) which the client would like ToolBook to execute. ToolBook's default response to this message is to execute the statement execute <*command*>.

Examples

The following example demonstrates how the remoteCommand message can be used.

```
--This example prevents a client application from shutting down this
```

```
--instance of ToolBook by sending an exit message.  All other commands
--are accepted.
to handle remoteCommand command
    if command contains "send exit"
        respondRemote "Failed: Denied"
    else
        forward
    end
end
```

Chapter 1

Writer's addendum

Change log

10/26/90	First draft. JDD
11/1/90	Incorporated extensive revisions. JDD
11/2/90	Entered in original edits and put into APS formats. SLD
11/5/90	Changed Postscript rules back to regular lines. SLD
11/5/90	Incorporated new edits and fixed mistakes in editing. JDD