DDELIB 1.3
Horizon Technologies Inc.

Introduction:

The Dynamic Data Exchange (DDE) Dynamic Link Library (DLL) offers Microsoft Windows programmers a tool that reduces the effort required to add DDE support to their applications.

DDE is a protocol for sending data and commands between two applications in real time. This link makes it possible to build an integrated system from specialized component programs. For example, a DDE session could be established to transfer numerical data from a spreadsheet to a word processing program. By means of the DDE protocol the spreadsheet could automatically send updated data to the word processor every time the user changed an entry in the spreadsheet.

As attractive as this sounds, DDE has a major drawback - it is very difficult for programmers to implement. Not only must they deal with the data being transferred to and from the application being developed, but also with the difficult and confusing DDE protocol itself.

We cannot offer much assistance in the processing of data, as this is inherent to the structure of every program. However, there is considerable relief from the burdens of the DDE protocol in the form of this library. **The DDE Library provides a simplified means of implementing the DDE protocol in any Windows application**.

Overview:

This overview summarizes the function calls available in the DDE Library. A general knowledge of the DDE protocol can be obtained from Microsoft documents: Chapters 8, 9 and 10 of the Microsoft Windows Software Development Kit - Windows Extensions - Version 2.0, and chapter 22 of the Microsoft Windows Guide to Programming for Windows 3.0 will provide the needed background.

The DDE Library tracks all DDE messages. The application, whether a client, a server or both, does not have to respond to any DDE messages in its window procedure. The DDE Library will create special windows to handle all of the DDE message traffic. The application needs only to interact with the library functions provided.

The DDE Library handles all message flow control and acknowledgements. It tracks the current state of the DDE conversation and sends messages only when permitted by the DDE protocol. The application does not have to determine whether or not to acknowledge a received message or wait until an acknowledgement is received from a transmitted message.

The DDE Library handles all atom creation and deletion. It will convert a string to an atom and back again so the application does not need to work with any of the atom manager functions.

The DDE Library handles much of the memory management required for DDE. Any memory management required by the application is documented by those functions requiring it.

The DDE Library provides special support services to help DDE server applications track information relevant to DDE, but not necessarily part of the DDE protocol itself.

Applications requiring DDE client functionality use the following functions:

DDEInitiate and DDETerminate for session management.

DDEAdvise and DDEUnadvise to establish and close advise circuits.

DDERequest and DDEPoke for one time data transfers from and to the server respectively.

DDEExecute to send a command to the server.

Applications requiring DDE server functionality use these functions:

DDERegisterTopic to provide a service to other potential client applications.

DDEGetSessionAppAtom, DDEGetSessionAppName, DDEGetSessionTopicAtom and DDEGetSessionTopicName to obtain information about the client application.

DDESendData to transmit data to client applications.

DDERegisterAdvise, DDEGetAdvise and DDEUnregisterAdvise to record and recall outstanding advise circuits.

DDEUnregisterTopic to remove a particular service.

Each of the functions of the DDE Library are described in detail on the following pages. The information provided includes the function description, syntax, parameter list, return value, and helpful comments. These descriptions also document the format of any callback functions that may be required. An appendix is supplied that describes the different structures used.

DDE Library can communicate with a variety of DDE based applications including Microsoft's Excel and Word for Windows. DDE Library also comes with sample programs that demonstrate the use of the library.

Syntax:

HWND DDEInitiate (hOwner, lpszApp, lpszTopic)

Client applications use this function to initiate a DDE session with the first server that responds. This DDE function must be called before any of the other client DDE functions.

Parameter	**Type/Description**
hOwner	**HWND** Handle to the top level window of the application.
lpszApp	**LPSTR** Points to a character string that names the requested application. The string must be a null terminated character string.
lpszTopic	**LPSTR** Points to a character string that names the requested topic. The string must be a null terminated character string.

Return Value:

A handle to the initiated DDE session which is used in subsequent client DDE function calls. It is NULL if the function is unsuccessful.

Syntax:

WORD DDEAdvise (hClient, lpszItem, lpAdvise, lpfnAdviseCallBack)

Client applications use this function to request the establishment of an advise circuit for a particular data item. Whenever the server determines that the value of the data item has changed, it will notify the client by calling the *lpfnAdviseCallBack* function. This function returns after the server has acknowledged the advise.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
lpszItem	**LPSTR** Points to a character string that names the requested item. The string must be a null terminated character string.
lpAdvise	**DDEADVISE FAR *** Points to a DDEADVISE structure that specifies the particular options for this advise circuit. See *Advise Structure* below.
lpfnAdviseCallBack	**DDECALLBACK** Is the procedure instance of the callback function that will be invoked each time the server issues an advise. See *Callback Function* below.

Return Value:

A word in the format of the DDEACK structure. See Appendix for more information.

Advise Structure:

Set *fDeferUpd* to TRUE to cause the server to exclude the DDEDATA structure as part of the data message. Set *fAckReq* to TRUE to cause the server to wait for an acknowledgement before sending another data message. Set *cfFormat* to a valid clipboard format, such as CF_TEXT, identifying the format of the data requested. See Appendix for more information.

Never set both *fAckReq* and *fDeferUpd* to TRUE, since the server would have no way of letting the client know whether or not ot should acknowledge a particular data message.

Comments:

Setting *fDeferUpd* to TRUE provides a mechanism to avoid frequent data transfers if only a notification is required.

Note that when setting *fAckReq* to TRUE the server will not receive an acknowledgement until the callback function returns.

Callback Function:

WORD FAR PASCAL *Advise* (hClient, iMessage, lpszItem, hData)

Advise is a placeholder for the application-supplied function name. The actual name must be exported by including it in an **EXPORTS** statement in the application's module definition file.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
iMessage	**unsigned** A value indicating why the callback function was invoked. See *messages* below.
lpszItem	**LPSTR** Points to a character string that names the item whose value is currently being advised. The string is a null terminated character string.
hData	**HANDLE** A handle to a global memory object in the form of a DDEDATA structure for DDE_DATA messages. It will be NULL if the *fDeferUpd* flag of the DDEADVISE strucure passed to *DDEAdvise* was TRUE. It is also NULL for all other values of *iMessage*. See *Data Structure* below. Do not free this memory object.

Messages:

The messages sent to the callback function contained in *iMessage* are as follows:

Value	**Description**
DDE_ACK	First message sent when the circuit opens successfully. *hData* is NULL.
DDE_DATA	One message sent each time the server advises the client of the new data.
DDE_TERMINATE	Sent when the circuit is closing or if the circuit could not be established.

Return Value:

The return value, defined as a WORD, is unused.

Data Structure:

If the *fDeferUpd* flag in the DDEADVISE structure passed to the *DDEAdvise* function was TRUE then *hData* will be NULL. *cfFormat* should be the format set in the DDEADVISE structure passed to the *DDEAdvise* function.

Syntax:

WORD FAR PASCAL DDEUnadvise (hClient, lpszItem)

Client applications use this function to request the termination of an advise circuit for a particular data item previously established by a call to the *DDEAdvise* function. This function returns after the server has acknowledged the unadvise.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
lpszItem	**LPSTR** Points to a character string that names the requested item. The string must be a null terminated character string.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

Syntax:

WORD DDERequest (hClient, lpszItem, cfFormat, lpfnRequestCallBack)

Client applications use this function to request a particular data item. The server will respond to the client by calling the *lpfnRequestCallBack* function. This function returns after the server has acknowledged the request.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
lpszItem	**LPSTR** Points to a character string that names the requested item. The string must be a null terminated character string.
cfFormat	**WORD** should be set to a valid clipboard format, such as CF_TEXT, identifying the format of the data requested.
lpfnRequestCallBack	**DDECALLBACK** Is the procedure instance of the callback function that is invoked when the server responds with the data. See *Callback Function* below.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

Callback Function:

WORD FAR PASCAL Request (hClient, iMessage, lpszItem, hData)

Request is a placeholder for the application-supplied function name. The actual name must be exported by including it in an **EXPORTS** statement in the application's module definition file.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
iMessage	**unsigned** A value indicating why the callback function was invoked. See *messages* below.
lpszItem	**LPSTR** Points to a character string that names the item whose value is currently being requested. The string is a null terminated character string.
hData	**HANDLE** A handle to a global memory object in the form of a DDEDATA structure for DDE_DATA messages. Otherwise, it is NULL. See *Data Structure* below. Do not free this memory object.

Messages:

The messages sent to the callback function contained in *iMessage* are as follows:

Value	**Description**
DDE_DATA	Sent if the server fulfilled the request and is providing the data.
DDE_TERMINATE	Sent if the server could not fulfill the request and hence no data is available. *hData* is NULL.

Return Value:

The return value, defined as a WORD, is unused.

Data Structure:

cfFormat is the format requested in the *DDERequest* function.

Syntax:

WORD FAR PASCAL DDEPoke (hClient, lpszItem, hPoke)

Client applications use this function to send, unsolicited, a particular data item to the server. This function returns after the server has acknowledged the poke.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
lpszItem	**LPSTR** Points to a character string that names the item whose data is being sent. The string must be a null terminated character string.
hPoke	**HANDLE** A handle to a global memory object in the form of a DDEPOKE structure. It must have been allocated with *GlobalAlloc* using the GMEM_DDESHARE flag. See *Poke Structure* below.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

Poke Structure:

Set *fRelease* to TRUE to cause the server to free the memory object. Otherwise, it is the client application's responsibility to free the memory object. Set *cfFormat* to a valid clipboard format, such as CF_TEXT, identifying the format of the data being sent.

Comments:

Even if *fRelease* has been set to TRUE in the DDEPOKE structure, it is still the client application's responsibility to free the memory object if the poke fails. This can be determined by examining *fAck* in the return value. It will be FALSE if the poke failed.

Syntax:

WORD FAR PASCAL DDEExecute (hClient, hCommand)

Client applications use this function to send, unsolicited, a particular command to the server. This function returns after the server has acknowledged the command.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.
hCommand	**HANDLE** A handle to a global memory object which contains a null terminated ASCII string of commands that the server should execute. It must have been allocated with *GlobalAlloc* using the GMEM_DDESHARE flag. Example: [open ("foo.xlm")] [run ("r1c1")]. This memory object will be freed by the DDE Library.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

Syntax:

WORD FAR PASCAL DDETerminate (hClient)

Client applications use this function to terminate a session previously established *DDEInitiate*. This function invalidates the *hClient* handle which can no longer be used. This function returns after the server has acknowledged the command.

Parameter	**Type/Description**
hClient	**HWND** Handle to a DDE session returned by *DDEInitiate*.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

Comments:

Advise circuits established with *DDEAdvise* do not need to be explicitly closed with a call to *DDEUnadvise* before terminating a session.

Syntax:

HWND DDERegisterTopic (hOwner, lpszApp, lpszTopic, lpfnTopicCallBack)

Server applications use this function to register its services for a particular topic. Any potential client application can establish a session with this server by issuing a *DDEInitiate* for this particular application and topic.

Parameter	**Type/Description**
hOwner	**HWND** Handle to the top level window of the application.
lpszApp	**LPSTR** Points to a character string that names the name of the application providing the service. The string must be a null terminated character string.
lpszTopic	**LPSTR** Points to a character string that names the name of the topic that the application is servicing. The string must be a null terminated character string.
lpfnTopicCallBack	**DDECALLBACK** Is the procedure instance of the callback function that is invoked each time a client requests a service. See *Callback Function* below.

Return Value:

A handle to the created DDE topic server which is used in subsequent server DDE function calls. It is NULL if the function is unsuccessful.

Comments:

A server may register more than one topic by repeatedly calling *DDERegisterTopic*. Each invocation should (but is not required to) use the same value for *lpszApp*.

The application can use the same callback function for more than one topic. Four functions: *DDEGetSessionAppAtom, DDEGetSessionAppName, DDEGetSessionTopicAtom,* and *DDEGetSessionTopicName* can be used within the callback function to determine the application and topic that a client is communicating with.

Callback Function:

WORD FAR PASCAL Topic (hSession, iMessage, lpszItem, hData)

Topic is a placeholder for the application-supplied function name. The actual name must be exported by including it in an **EXPORTS** statement in the application's module definition file.

Parameter	**Type/Description**
hSession	**HWND** Handle that identifies a particular DDE session with a client. This is not the return value from *DDERegisterTopic*.
iMessage	**unsigned** A value indicating why the callback function was invoked. See *messages* below.
lpszItem	**LPSTR** Points to a character string that names the item of interest. This may be NULL for some messages. The string is a null terminated character string.
hData	**HANDLE** A handle to a global memory object in a form dependant on the message. Do not free this memory object.

Return Value:

The return value is defined as a WORD. For messages that require a return value, fill this word using the format of the DDEACK structure. The DDEACK structure is defined in the Appendix. Each message value documented below defines whether or not it requires a return value.

Messages:

The messages sent to the callback function contained in *iMessage* are as follows:

Value	**Description**
DDE_INITIATE	Message sent when a new client establishes a session. *lpszItem* and *hData* are NULL. The return value is not used.
DDE_TERMINATE	Message sent when a session with a client is being terminated. *lpszItem* and *hData* are NULL. The return value is not used.
DDE_POKE	Message sent when the client is offering data for a particular item. *lpszItem* contains the name of the item. *hData* is in the form of a DDEPOKE structure. Make sure to set *fAck* and *fBusy* in the return value since it is used.
DDE_REQUEST	Message sent when the client is requesting a one time data transfer for a particular item. *lpszItem* contains the name of the item. *hData* is in the form of a DDEADVISE structure.

	Check *cfFormat*. No need to check *fRelease* or *fAckReq*. Do not free this memory object. Make sure to set *fAck* and *fBusy* in the return value since it is used. If *fAck* is TRUE the server is required to send the data using *DDESendData*.
DDE_EXECUTE	Message sent when the client is requesting the execution of a command string. *lpszItem* is NULL. *hData* is simply a null terminated ASCII string. Make sure to set *fAck* and *fBusy* in the return value since it is used.
DDE_ADVISE	Message sent when the client is requesting a continuous data transfer for a particular item whenever its value changes. *lpszItem* contains the name of the item. *hData* is in the form of a DDEADVISE structure. Check *fAckReq, fDeferUpd,* and *cfFormat*. Do not free this memory object. Make sure to set *fAck* and *fBusy* in the return value since it is used. If *fAck* is TRUE the server should send advise data using *DDESendData*.
DDE_UNADVISE	Message sent when the client is requesting a continuous data transfer to be terminated for a particular item. *lpszItem* contains the name of the item. *hData* is NULL. Make sure to set *fAck* and *fBusy* in the return value since it is used.

Syntax:

ATOM DDEGetSessionAppAtom (hSession)

Server applications use this function to determine the application name that the client has used to communicate with it. This is useful if the same callback function has been passed to multiple calls to *DDERegisterTopic*.

Parameter	**Type/Description**
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*.

Return Value:

An atom value (integer) representing the application name. Use *GlobalGetAtomName* or *DDEGetSessionAppName* to retrieve the actual string.

Syntax:

int DDEGetSessionAppName (hSession, lpszApp, nSize)

Server applications use this function to determine the application name that the client has used to communicate with it. This is useful if the same callback function has been passed to multiple calls to *DDERegisterTopic*.

Parameter	**Type/Description**
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*.
lpszApp	**LPSTR** Points to a character string to receive the name of the application. The string will be a null terminated character string.
nSize	**int** Specifies the maximum size (in bytes) of the buffer pointed to by *lpszApp*.

Return Value:

The return value specifies the actual number of bytes copied to the buffer.

Syntax:

ATOM DDEGetSessionTopicAtom (hSession)

Server applications use this function to determine the topic name that the client has used to communicate with it. This is useful if the same callback function has been passed to multiple calls to *DDERegisterTopic*.

Parameter	**Type/Description**
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*.

Return Value:

An atom value (integer) representing the topic name. Use *GlobalGetAtomName* or *DDEGetSessionTopicName* to retrieve the actual string.

Syntax:

int DDEGetSessionTopicName (hSession, lpszTopic, nSize)

Server applications use this function to determine the topic name that the client has used to communicate with it. This is useful if the same callback function has been passed to multiple calls to *DDERegisterTopic*.

Parameter	**Type/Description**
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*.
lpszTopic	**LPSTR** Points to a character string to receive the name of the topic. The string will be a null terminated character string.
nSize	**int** Specifies the maximum size (in bytes) of the buffer pointed to by *lpszTopic*.

Return Value:

The return value specifies the actual number of bytes copied to the buffer.

Syntax:

HWND DDESendData (hSession, iMessage, lpszItem, hData)

Server applications use this function to send data to a client application in direct response to a request message or as a result of an advise circuit having been established. This function returns after the client has acknowledged the data message if the *bAckReq* field in the DDEDATA structure is TRUE. Otherwise it will return immediately after the message is sent.

Parameter	**Type/Description**
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*.
iMessage	**unsigned** A value indicating the reason the data is being sent to the client. Set this value to DDE_REQUEST if the data is in response to a request message, otherwise set it to DDE_ADVISE.
lpszItem	**LPSTR** Points to a character string that names the item whose data is being sent. The string must be a null terminated character string.
hData	**HANDLE** A handle to a global memory object in the form of a DDEDATA structure. It must have been allocated with *GlobalAlloc* using the GMEM_DDESHARE flag. See *Data Structure* below.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

Data Structure:

If this data is sent in response to an advise message, set *fAckReq* to the same value as the *fAckReq* field of the DDEADVISE structure sent previously by the client. If this data is sent in response to a request message *fAckReq* may be set to either TRUE or FALSE. See the appendix for more information. Set *fRelease* to TRUE if the client should free the memory object. Otherwise, it is the server application's responsibility to free the memory object. Set *cfFormat* to a valid clipboard format such as CF_TEXT.

Never set both *fAckReq* and *fRelease* to FALSE, since it would be the server applications responsibility to free the memory object but it would not have a way of knowing when the client application had completed processing the data.

Comments:

Even if *fRelease* has been set to TRUE in the DDEDATA structure, it is still the server application's responsibility to free the memory object if the send fails. This can be determined by examining *fAck* in the return value. It will be FALSE if the send failed.

Syntax:

HANDLE DDERegisterAdvise (hList, hSession, lpszItem, fAckReq, fDeferUpd, cfFormat)

Server applications use this function to record information about a new advise circuit in response to an advise message. This function appends the new advise circuit information to the list indicated by hList.

Parameter	Type/Description
hList	**HANDLE** Handle that identifies the list of advise circuits that this new advise should be appended to. This is the return value from a previous call to *DDERegisterAdvise* or *DDEUnregisterAdvise*. If this parameter is NULL a new list will be created. Upon return from this function, the value of *hList* will no longer be valid and should be replaced by the return value.
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*.
lpszItem	**LPSTR** Points to a character string that names the item of the advise circuit. The string must be a null terminated character string.
fAckReq	**int** This should be set to the *fAckReq* field in the DDEADVISE structure passed to the callback function registered by *DDERegisterTopic*.
fDeferUpd	**int** This should be set to the *fDeferUpd* field in the DDEADVISE structure passed to the callback function registered by *DDERegisterTopic*.
cfFormat	**int** This should be set to the *cfFormat* field in the DDEADVISE structure passed to the callback function registered by *DDERegisterTopic*.

Return Value:

A handle to the newly created advise list in the case that *hList* is NULL, and a handle to the modified advise list in the case that *hList* was not NULL.

Comments:

Make sure not to re-use the value of *hList* but to use the return value from the function instead.

Syntax:

HANDLE DDEGetAdvise (hList, nItem, lphSession, lpszItem, nItemLen, lpfAckReq, lpfDeferUpd, lpcfFormat)

Server applications use this function to recall information about outstanding advise circuits from the list indicated by hList. This is useful to check if any clients have registered an advise circuit on an item that the server application has determined to have changed. The return information is useful for calling *DDESendData*.

Parameter	**Type/Description**
hList	**HANDLE** Handle that identifies the list of advise circuits to scan. This is the return value from a previous call to *DDERegisterAdvise* or *DDEUnregisterAdvise*.
nItem	**int** Index number of the advise circuit in *hList*. To scan through all entries, repeatedly call this function with successively larger integer values for *nItem*, beginning at zero, until the return value indicates that the item does not exist.
lphSession	**HWND FAR *** Pointer to variable to receive the *hSession* value previously passed to *DDERegisterAdvise*.
lpszItem	**LPSTR** Pointer to variable to receive the *lpszItem* string previously passed to *DDERegisterAdvise*.
nItemLen	**int** Specifies the maximum size (in bytes) of the buffer pointed to by *lpszItem*.
lpfAckReq	**int FAR *** Pointer to variable to receive the *fAckReq* value previously passed to *DDERegisterAdvise*.
lpfDeferUpd	**int FAR *** Pointer to variable to receive the *fDeferUpd* value previously passed to *DDERegisterAdvise*.
lpcfFormat	**int FAR *** Pointer to variable to receive the *cfFormat* value previously passed to *DDERegisterAdvise*.

Return Value:

The return value specifies the actual number of bytes copied to the buffer. It is -1 to indicate that no advise entry exists for the specified *nItem*.

Comments:

As items are added and removed with *DDERegisterAdvise* and *DDEUnregisterAdvise* the item number for a particular advise will not remain constant.

Syntax:

HANDLE DDEUnregisterAdvise (hList, hSession, lpszItem)

Server applications use this function to delete recorded information about a terminated advise circuit in response to an unadvise message. This function removes the advise circuit information from the list indicated by hList.

Parameter	Type/Description
hList	**HANDLE** Handle that identifies the list of advise circuits that this advise should be removed from. This is the return value from a previous call to *DDERegisterAdvise* or *DDEUnregisterAdvise*. Upon return from this function, the value of *hList* will no longer be valid and should be replaced by the return value.
hSession	**HWND** Handle that identifies a particular DDE session with a client identified as the first parameter of the callback function registered by *DDERegisterTopic*. This is not the return value from *DDERegisterTopic*. If this value is NULL, all advise circuits recorded in *hList* will be removed and the list will be freed.
lpszItem	**LPSTR** Points to a character string that names the item of the advise circuit. The string must be a null terminated character string. If this value is NULL, all advise circuits recorded in *hList* that belong to the specified *hSession* will be removed.

Return Value:

A handle to the modified advise list.

Comments:

Make sure not to re-use the value of *hList* but to use the return value from the function instead.

Syntax:

HWND DDEUnregisterTopic (hServer)

Server applications use this function to unregister its services for a particular topic. This function will first terminate all currently active sessions with clients before returning.

Parameter	**Type/Description**
hServer	**HWND** Handle to a DDE topic server established with *DDERegisterTopic*.

Return Value:

A word in the format of the DDEACK structure. See Appendix.

The DDEACK structure:

A word in the format of the DDEACK structure is returned by all of the DDE library functions except *DDEInitiate* and *DDERegisterTopic*. The fields that are packed into this word are described in the table below.

Field	**Type/Description**
fAck	**unsigned:1** Boolean flag that is TRUE for a positive acknowledgement and FALSE for a negative (nack) acknowledgement.
fBusy	**unsigned:1** Boolean flag that indicates the server could not process the request because it is currently busy. This field only has meaning only if *fAck* is FALSE.
bAppReturnCode	**unsigned:8** Can be set to any value desired for further interprocess status communications. However, the DDE library functions also set this return code to indicated failures on their part as well. See the table below.

The following table represents the values that the DDE library functions may supply in the *bAppReturnCode* of the DDEACK structure.

Value	**Meaning**
DDE_OK	Function completed ok.
DDE_NACK	A nack was received.
DDE_BADHANDLE	The session handle passed to the function is invalid.
DDE_BADSTATE	The current state of communications can not perform the action requested by this function.
DDE_MEMORY	A memory allocation error occurred during the processing of this function.
DDE_BADITEM	The item passed to the function is invalid.
DDE_BADTOPIC	The topic passed to the function is invalid.

The DDEADVISE Structure:

The DDEAdvise structure, sent to the server from the client, contains the following fields.

Field	Type/Description
fAckReq	**unsigned:1** Boolean flag that is TRUE if the server is required to wait for an acknowledgement to be returned before sending another advise message. The *fAckReq* field in the DDEDATA structure sent as part of the data message to the client must be set equal to this field by the server.
fDeferUpd	**unsigned:1** Boolean flag that indicates the server should not send the DDEDATA structure when sending a data message. It is assumed that the client will send a request message to the server when it requires the actual data.
cfFormat	**int** A valid clipboard format, such as CF_TEXT, identifying the format of data requested.

The DDEDATA Structure:

The DDEData structure, sent to the client from the server, contains the following fields.

Field	Type/Description
fAckReq	**unsigned:1** Boolean flag that is TRUE if the client is required to send an acknowledgement back to the server.
fRelease	**unsigned:1** Boolean flag that is TRUE if the client is expected to free the global memory object.
fResponse	**unsigned:1** Boolean flag that is TRUE if the data message is in response to a request. FALSE if the server initiated the transfer as a result of an established advise circuit.
cfFormat	**int** A valid clipboard format, such as CF_TEXT, identifying the format of data sent.
Value	**BYTE[n]** The data.

The DDEPOKE Structure:

The DDEPoke structure, sent to the server from the client, contains the following fields.

Field	Type/Description
fRelease	**unsigned:1** Boolean flag that is TRUE if the server is expected to free the global memory object.
cfFormat	**int** A valid clipboard format, such as CF_TEXT, identifying the format of data sent.
Value	**BYTE[n]** The data.