Dynamic Data Exchange (DDE) and NetDDE FAQ

Why it is not replaced by COM, how it works, network DDE, links to other sources of information

Before you leave ---

Click the links to the left to investigate our software. Most of it is free!

- **On this page you will find**
  - Overviews and detailed articles about DDE and how it used
  - Download links for DDE tools, components and samples
  - Where to find DDE information on MSDN
  - Newsletter signup, tell you friends about this site
- **Related pages**
  - Support for our DDServer and DDClient ActiveX controls
  - DDE in Visual Basic, Web Browsers, the Windows Shell, Excel and other applications

Questions and topics

- **DDE, COM and interprocess communication**
  - What is DDE and why is it totally different from COM?
  - When is DDE more suitable?
  - When is COM more suitable?
  - COM and DDE data transfer speeds compared
  - Will Microsoft continue to support DDE?
  - Other forms of interprocess communication
- **Programming DDE**
  - DDE mechanism overview
  - Network DDE (NetDDE)
  - Using the DDEML Application Programming Interface
  - A memory leak using DDEML
  - Why do I get the DDEML Reentrancy error?
  - Why does DDE sometimes change the case of the Service, Topic and Item names?
  - Service, Topic and Item names: number and length limits
  - Why does my Execute command data get corrupted?

DDE Software download area

Software Click on the file name to download	**File**	**File size KB**
DDClient ActiveX control using the VB5 runtime	ddclv305.zip	567
DDClient ActiveX control using the VB6 runtime	ddclv306.zip	547

DDServer ActiveX control using the VB5 runtime	ddsvv105.zip	452
DDServer ActiveX control using the VB6 runtime	ddsvv106.zip	424
WebSpy browser URL monitor for IE and Netscape Navigator with source code	webspy.zip	498
Sample C program to control Excel	xlddec.zip	14
Sample Visual Basic program to control Excel. Requires DDClient	xlddevb.zip	13
DDStudio DDE utility and diagnostic tool	ddssetup.exe	494
Sample native Visual Basic DDE server and client	vbdde.zip	16
Sample program using DDEML in an MFC project	rddeclnt.zip	180
The 16-bit DDESHARE for managing NetDDE in Windows 9x	ddes311.zip	5

What is DDE and why is it totally different from COM?

DDE stands for Dynamic Data Exchange. That's exactly what it does, and nothing more. It sends data between applications using Windows messages according to a documented protocol. Saying that DDE is old-fashioned and is being replaced by COM is something you see repeated parrot fashion over and over again. DDE and COM do not work in the same way and they solve different problems. Here are some points of difference:

- COM is synchronous, one party makes subroutine calls into the other and must wait until the call returns. If the called component is busy the caller is blocked until it becomes free. DDE is asynchronous, a well-programmed client sends a Windows message to the server and carries on processing. Windows holds the message and sends it to the server when the server is ready to process it.
- COM is complex to program but powerful. The client can manipulate

objects within the server as if they belonged to the client. DDE is straightforward to implement but all it can do is transmit data. It can only control another application because the recipient can treat data as a command.

- COM works well when the client creates an instance of a server object or application for its own use. Programming a continuously running server, to which clients can attach when they wish, is possible but tricky. DDE of itself is incapable of creating objects (although OLE1 used it as a transport mechanism). The essence of DDE is that clients attach to an already running server.
- Applications using COM almost always need support DLLs such as the VB runtimes or MFC. Programs using DDE can be mean and lean and do not need extra support DLLs.
- COM interfaces are tightly specified and contained within the software component. Some documentation is normally built into the interface. The user of a COM component knows exactly which methods are available and how to call them. DDE interfaces are specified only in external documentation.
- Because of the tight interface specification, upgrading COM components can be a nightmare. With DDE the server or client can be changed independently.
- COM is frequently used to communicate with a DLL in the same process space (called an in-process server). Such as server is often termed an ActiveX control and has the extension OCX. Our DDClient and DDServer Visual Basic components are in-process servers. Using DDE to communicate between components of the same application is possible but has no benefits.
- COM communication with a remote machine is called DCOM (Distributed COM). If you make a DCOM call to a remote machine which does not respond, your program (or at least the thread making the call) is stuck. DCOM has a fixed built-in timeout which you cannot change. DDE with a remote machine is called NETDDE, it is used by the Hearts game and Chat which come with Windows. Timeout is controlled by the program.
- Under Windows 3.1 and 9x it is possible to crash the system by badly programmed DDE, because the message queue can be filled. NT does not crash in this way, COM does not suffer from the problem at all.
- DDE is totally "Bit blind", neither the client nor server can tell whether the other application is 16-bit or 32-bit. Indeed, the server cannot know whether the client is on the same computer or not. Connecting 16 to 32 bit COM components is not usually possible.

It is easy to see from the above points why a DDE server is the most widely used way of providing data obtained from any form of hardware interface. In

particular:

- The server can run continuously.
- It will probably serve a wide range of clients but can be updated without requiring them all to be recompiled.
- It is immune from the problems associated with different versions of the system DLLs on different machines.
- It can be small and self-contained.
- Clients cannot interfere with the running of the server by being slow or busy.

Related topics -
When is DDE more suitable?
When is COM more suitable?
DDE and COM in Visual Basic
The COMclean registry cleaner page has an overview of the Registry entries associated with COM.

Question and topic list DDE Software downloads Home page

When is DDE more suitable?

When using COM is impossible

- **Connecting to a running instance of a program which does not register in the Running Object Table (ROT).**
  For example, ActiveX server applications written in Visual Basic do not register themselves in the ROT. (It is possible by a piece of 'Black Belt' programming, described here.)

- **Connecting to a particular instance of an application**
  Some applications such as Excel do register themselves in the ROT and support multiple instances. However, there is no way to connect to a particular instance of Excel with COM. For example, the GetObject() method in Visual Basic returns a random instance, even when a file name is specified.

  With DDE you can manipulate any worksheet without regard to the instance hosting it, because Excel offers a separate topic name for each worksheet. For more information on controlling Excel go to Controlling Excel with DDE.

With some care it is possible to connect to particular instances with DDE, even when the topics supported by each instance are the same. The client must start when at most one of the instances of the server is running. The client must capture the instance handle of each instance as it registers with DDEML, and connect using it. Connections made using a string for the service name, e.g. IExplore, will connect to a random instance.

Writing compact programs, or when the compiler does not make COM easy

The amount of code required to operate a DDE interface is trivial compared to that required for COM. In the DDE Software download area there are sample programs (with full source) which manipulate Microsoft Excel using DDE. The compiled C program is only 17kb, and it does not rely on any external DLLs (apart from the Common Controls which is for the status bar, not DDE). The Visual Basic example uses the Evaluation version of DDClient, so you can modify the project. VB does make COM easy, but if you need to hook up to a running instance of Excel, DDE is the only way you can be certain of connecting to the required worksheet.

When the server is a self standing program which must execute whether or not is has clients

In other words, when the server is the primary application or must have priority on processor time. Hardware monitoring is an example of this situation. The overhead of queuing a message is fixed and the process cannot hang. Quite apart from the complexity of programming COM events, the server must wait for each client to process the data, and its performance therefore depends upon the performance of the clients. If just one of the clients hangs in the COM (or DCOM) event routine the server, or at least one thread, is brought down with it.

A DDE server is the most widely used way of providing data obtained from any form of hardware interface because:

- The server can run continuously.
- It will probably serve a wide range of clients but can be updated without requiring them all to be recompiled.
- It is immune from the problems associated with different versions of the system DLLs on different machines.
- It can be small and self-contained.
- Clients cannot interfere with the running of the server by being slow or busy.

Networked DDE is suitable when a server is supplying data to a number of clients on different machines. Messages are queued within the client machines and do not affect server performance. NetDDE depends on NetBIOS which can use TCP/IP, so NetDDE works over the Internet.

When the complexity and tight specification of COM, and the requirement to use GUIDs would be a hindrance rather than a help.

An example is the way the Windows shell launches programs, using DDE to pass a filename or command. COM would be gross overkill for this simple task. For more information see the topic How the Windows Shell uses DDE and the File Association Editor page.

For program monitoring

DDE is ideal for allowing one program to monitor another, because neither program ever executes in the context of the other so they cannot interfere with each other. Internet Explorer and Netscape Navigator provide a DDE interface for monitoring the URLs they load. You can see this in action with our Webspy freeware browser spy. It is a good example of an application for which DDE is much more suitable than COM (ActiveX). WebSpy can run without a browser running. WebSpy or the browser(s) can be started first, WebSpy does not interfere with browsing in any way, either program can be terminated at any time without affecting the other. It is written in Visual Basic and uses the DDClient and DDServer components. It can be downloaded from the Software download area

Related topics -
What is DDE and why is it totally different from COM?
When is COM more suitable?
DDE and COM in Visual Basic

Question and topic list DDE Software downloads Home page

When is COM more suitable?

Whenever the client is the primary application and is in a position to create an instance of an object or application for its own use. The server is dedicated to processing on behalf of the client.

This scenario applies to many office situations, for example a Visual Basic program creating an Excel spreadsheet or Word creating a document which contains images created by another application. Before COM, DDE was used to instruct the server which manipulations to perform and DDE was used to transmit the results back to the client. For this purpose Microsoft have quite rightly extended the COM interfaces and frozen the DDE ones. They have for the most part been left in place for backward compatibility, and to allow the applications to gather data from DDE only servers.

For information on programming Microsoft Office components using COM go to http://www.microsoft.com/officedev/articles/OffObjPr.htm.

Related topics -
What is DDE and why is it totally different from COM?
When is DDE more suitable?
DDE and COM in Visual Basic

Question and topic list DDE Software downloads Home page

COM and DDE data transfer speeds compared

The transfer rates in the first group are estimated ball-park figures for 300MHz machines driven as fast as possible. We accept no responsibility for their accuracy, your mileage may vary widely. They are based on a small number of tests, the tests were not checked to ensure comparing like with like, there were only a small number of measurements. Remember that Windows can queue DDE messages, but COM is synchronous and the caller must wait for the other application to finish processing the call.

- Sending data to a client on an advise loop using DDEML - 1000 transactions/sec.
- Sending data to a client on an advise loop using our DataSend control - 2000 transactions/sec.
- Sending data to a client on an advise loop using raw DDE - 2000 transactions/sec.
- Sending data by NetDDE with DDEML - 1300 transactions/sec. (The server and client can process in parallel.)
- DDE transactions in Visual Basic using the built-in mechanism 100 - transactions/sec.

- Out-of-process COM calls - 2000-3000 transactions/sec.
- DCOM calls are slower than out-of-process, similar to NetDDE.

The following numbers are taken from the COM tutorial at DevelopMentor for 100 (lower figure) and 4 (higher figure) bytes of parameters. They are slower than our figures. DevelopMentor say the numbers are fairly old, this probably means that they were measured on a slower machine.

- In-process COM calls - 119,240-129,041 transactions/sec.
- Out-of-process COM calls - 968-1765 transactions/sec.
- DCOM calls (LAN) -- 342-495 transactions/sec.
- DCOM calls (WAN) -- 16-19 transactions/sec.

Do not be mislead by figures quoted for in-process COM calls, an in-process server is called in a similar way to a normal DLL. ActiveX which does not involve interprocess communication is much faster, as can be seen from the DevelopMentor table.

Question and topic list DDE Software downloads Home page

Will Microsoft continue to support DDE?

The DDE protocol has not changed since about 1990, indeed it would be disastrous if it did. Microsoft cannot stop supporting DDE, it's just messages passing between windows, which the Operating System relies on. The special bit is allowing applications to pass blocks of data, but that again is something which has to be done anyway so there is no cost in using the mechanism in DDE also.

What Microsoft has done is to stop updating the DDE interfaces of its applications in cases where the job is done better by COM. DDE itself will always be supported, the question is whether Microsoft will continue to support DDE communication with its applications. They will probably never remove the existing DDE interfaces for reasons of backwards compatibility, although changes and degradation have already happened.

What has not changed, and is most unlikely to, is the Shell DDE interface. DDE will continue to be used for launching documents when a file type has an associated application. It's so simple and quick to set up that the registry entries can be done by hand, although Explorer provides an excellent GUI for doing it. Using COM with its attendant GUID mechanism and complicated registry entries would be gross overkill. Similarly, the DDE

interface for managing program groups (the start menu in modern parlance) will probably be available for ever.

Question and topic list DDE Software downloads Home page

Other forms of interprocess communication

A list of methods of passing data between programs and which operating systems support them is to be found in Knowledgebase article Q95900 Interprocess Communication Mechanisms.

In additional to COM and DDE they are NetBIOS, Pipes, Sockets, Mailslots, Memory Mapped Files and WM_COPYDATA. Memory Mapped Files and WM_COPYDATA do not operate over a network. Only DDE and COM are mechanisms of the type in which multiple clients can connect to a single server. Named pipes are reported to be amongst the fastest ways of sending data between different computers.

Question and topic list DDE Software downloads Home page

DDE mechanism overview

Making the connection between a DDE server and a DDE client

DDE uses a hierarchy of three names, the SERVICE, the TOPIC and the ITEM. A DDE CONVERSATION is established using the service and topic names as a pair. It is convenient to name the pair a CHANNEL. It is roughly the equivalent of a telephone number. The item part of the name is used to identify the particular data or command being requested by the client once a conversation is established.

To establish a conversation a DDE client specifies the service/topic name pair (channel) it wishes to connect to. Windows broadcasts the request to all top level windows. The first server to accept is connected to the client and so a conversation is established.

The client may leave either the topic or the service name unspecified, or both. This is known as a WILDCONNECT. For example, if the topic name is not specified conversations may be established on all the topics supported by a server. If both are unspecified conversations may be established on all topics with all servers. It is up to the server whether to accept such a connection, and if so on what topics.

Unlike a telephone connection, any number of quite separate conversations may be in progress on the same channel, even between the same two applications. This is often done so that only one item of information or one type of command is handled by each conversation. There is no interaction at all between different conversations using the same pair of service and topic names.

Transactions within a DDE conversation

Just as the client application initiates the establishment of a conversation, it also initiates all the transactions. It can request data from the server as a once off (a REQUEST transaction), request being kept up to date about an item of data (an ADVISE or NOTIFY transaction), give commands to the server (an EXECUTE transaction) and send unsolicited data to the server (a POKE transaction). The client associates with all these transactions the item part of the identification. It informs the server of the data required by the client in a request transaction, the action to be taken by the server in an execute transaction or the data being passed to the server in a poke transaction.

It is also possible to use the item part of the name as the data itself, with the topic name indicating the context in which the data is to be used.

Raw DDE has no timing constraints except the order in which messages are sent. At the Windows message level, there is no distinction between synchronous and asynchronous transactions. Synchronous and asynchronous operation is a feature of the DDE management library, or DDEML.

To find out more go to How to get further information

Question and topic list DDE Software downloads Home page

Network DDE (NetDDE)

DDE across a network is simple to set up. It uses NetBIOS and since this can be run over TCP/IP, NetDDE can use the Internet. Given a fast connection, network DDE is if anything faster than between two programs on one machine, because the server and client can process in parallel. In order to operate DDE over a network the following steps are necessary.

1. On both machines, a DDE agent called NETDDE.EXE must be running, its job is to transmit the data across the network. Under Windows for Workgroups and Windows 9x NETDDE.EXE can be put in the Startup Group or be started manually. Under Windows NT it is a service which is started when required.

2. On the server side only, DDE shares must be created. These bind a service/topic name pair to a share name. For example, the pair WinChat/Chat has the share name Chat$. All the share information is stored in the registry, which has a different format under 3x/9x and NT. You may read that altering the registry by hand is required on Windows 9x, but this is not true.

The Windows API NDde makes the registry entries, it has calls such as NDdeShareAdd(). Application programs can use the API to share their DDE services automatically. The user creates shares manually with the program DDESHARE.EXE, which makes the necessary API calls.

In Windows for Workgroups 3.11 and Windows 9x only a 16-bit API is provided, so DDESHARE for those systems is a 16-bit program. If you do not have this program you can get it from the the DDE download area. The display of the program is not bug free. If it gets confused, which it may after deleting a share, exit and re-run. Under Windows NT only a 32-bit NDde API is available, so the NT version of DDESHARE is a 32-bit program.

The DDE server has no knowledge of the share and does not know if its clients are local or remote.

3. The client program must be changed to use the special service/topic pair \\computer-name\NDDE$/Share-name, for example service "\\RHAMINISYS\NDDE$" and topic "Chat$".

The client and server both hold a DDE conversation with NETDDE.EXE on the local machine. To make a connection, the client NETDDE sends the requested share name to the remote computer. The remote NETDDE agent looks in the registry for the share name. If it exists the permissions are checked. At the server end NETDDE then attempts to connect to the real service and topic names specified in the registry. The server may be started

automatically. Once the conversation has been established all DDE transactions are the same as between two programs on the same machine.

When connecting to NT the user may be prompted for the password and the number of sessions is limited. For details go to Windows NT NetDDE security and other issues.
See also the article How to get further information

Useful Microsoft Knowledgebase articles

This is not a complete list, you will be able to find more by searching the Microsoft site.

Q110315 NetDDE and RAS Connections
Q125703 Windows 95 support for Network DDE. Note that the NetDDE API referred to is the API for programmatically setting up DDE shares.
Q231337 Netdde.exe Does Not Relay WM_DDE_TERMINATE to Remote Clients. Microsoft claims this is fixed in SP5.
Q128491 Creating a NetDDE Link in Excel on NT

Question and topic list DDE Software downloads Home page

Using the DDEML Application Programming Interface

Many clients and servers use the DDEML Windows API. The same interface is available on all 16 and 32 bit versions of Windows. Raw DDE has to be programmed slightly differently on 16 and 32 bit systems, DDEML hides these differences. Using DDEML guarantees that the correct message protocol is followed, as it hides from the programmer the details of the messages and the fact that windows are created to send and receive them. DDEML based servers and clients make API calls to start transactions. The client may either have the call not return until a reply is received or a timeout has elapsed, or it may resume execution immediately and have a callback routine to process the reply when it arrives.

Raw DDE has no time dependence except the order in which messages must be sent. The DDEML API introduces timeouts, the client can tell DDEML how long it is prepared to wait. In a synchronous transaction, the client waits until the server has responded or the timeout has expired, then

processing continues. If the server does not respond within the timeout DDEML reports that the request has failed. Be careful to use only positive numbers as the timeout interval. -1 is the value which signifies an asynchronous transaction, the call returns immediately. Other negative number will be interpreted as large positive numbers.

In asynchronous operation, the client makes its request and processing continues immediately without waiting for a reply. The callback routine is entered when the server responds. The client can have any number of outstanding requests. It is up to the client program to implement any timeout. The program can instruct DDEML to abandon a transaction, in which case the callback will never be entered.

A server using raw DDE can easily delay sending a response to the client. When DDEML is used a reply must be generated and returned as the result of the callback routine. To achieve this a server application can return CBR_BLOCK for a request transaction. DDEML disables the server's callback function and also queues transactions that are sent by DDEML after its callback has been disabled. This feature gives the server an opportunity to gather data while allowing other applications to run in the system. When the server is ready it unblocks the callback and DDDEML makes the same request again.

You may find other documentation which states that only the affected conversation is blocked, however our testing shows that it is indeed the callback which is disabled. It is much better to refuse the request altogether, or to send back a special piece of data such as "N/A", which indicates to the client that the real data is not available yet.

DDEML requires a server application to register the service names that it supports. The names are broadcast to other applications which are using DDEML, which can then use the names to connect to the server. As well as the plain service name, DDEML provides an instance specific service handle. In other words, if two instances of the same application are started, a DDEML client can distinguish between them when establishing conversations.

There is a speed penalty of about 2 for using DDEML, when both server and client are doing almost nothing else. In the real world they both have to do other processing, so the speed penalty is far outweighed by the benefits

For 16-bit applications the API is implemented in a dynamic linked library called DDEML.DLL (DDE management library). The 32-bit API is implemented in User32.dll, an integral part of the Windows operating system. A discussion of the consequences can be found in the topics Why

does my program run differently on Windows 9x and NT? and What are the 16/32-bit and Windows version issues?.

To find out more go to How to get further information

Microsoft Knowledgebase articles

Q108933 Top 10 DDEML support issues
Q170626 Memory Leak in Global Shared Memory.

Question and topic list DDE Software downloads Home page

A memory leak using DDEML

There seems to be a problem with the **HDATA_APPOWNED** attribute of memory allocated by DdeCreateDataHandle. DDStudio at one time kept data in a block with this attribute, ready to send to any client that asked for it. Each time the data changed the old block was freed and a new one allocated.

The result was a rapid usage of memory, far in excess of the memory actually allocated by the calls to DdeCreateDataHandle. Hours of examining the code revealed no errors. The problem was solved by using GlobalAlloc instead, and creating a new system owned data handle with DdeCreateDataHandle for each request.

A Microsoft Knowledgebase article

Q170626 Memory Leak in Global Shared Memory. This problem is not confined to NT in our experience. It is best not to use the APPOWNED flag when creating DDEML data objects.

Question and topic list DDE Software downloads Home page

Why do I get the DDEML Reentrancy error?

The DDEML library is not designed to allow reentrancy of the callback routine. In other words, it must return before DDEML calls it again. It will be reentered if you yield to Windows and there is a DDE message for you on the queue.

It is therefore an absolute rule in programming your callback routine, do not yield to Windows.

Question and topic list DDE Software downloads Home page

Why does DDE sometimes change the case of the Service, Topic and Item names?

The Service, Topic and Item names in DDE are case insensitive because Global String Atoms are used, not the strings you type. It is not possible for a program to force the case and a bad bug to require one.

For all DDE names, the program gets a string handle to represent the string. The system does a lookup to see if the string has been allocated an Atom already. If so, it returns the existing value. Whoever requested the string Atom first determines the case. It follows that if you request a string handle for a string, then request the string corresponding to the handle, you will get back the string with an essentially random case.

If you are having problems with a program which demands a particular case, you may be able to resolve the situation by rebooting the machine. This clears out the global atom table. Then do some DDE using the correct case, which should stick until the last user frees their reference to the string.

Question and topic list DDE Software downloads Home page

Service, Topic and Item names: number and length limits

Because the Service, Topic and Item names are represented by String

Atoms, they are subject to the same limits as String Atoms themselves. The maximum length of strings is 255 characters and the case of letters is immaterial, as discussed above.

If you look at VC help files or MSDN on CDROM you will probably read the alarming fact that an Atom Table has a maximum of 37 entries. This is incorrect and has been remedied in the web version of MSDN, it is the number of buckets in the hash table. The maximinum number of different strings it can hold is 16384.

Question and topic list DDE Software downloads Home page

Why does my Execute command data get corrupted?

DDEML automatically translates execute strings. DDEML bases its translation decision on how DDEML was called, either by DdeInitiateA or DdeInitiateW when the application initialised DDEML. The way to avoid this problem is to use a Poke command instead of an Execute command to send data to a DDE server.

Question and topic list DDE Software downloads Home page

Using DDEML in an MFC project

Calling the DDEML library directly from within an MFC project

The sample DDE client project you can download here was written as a learning exercise. The original is 16-bit, but it compiles and runs under 32-bit as well (although the 32-bit version GPFs on exit). It is made available solely as a sample, absolutely without warranty, for your education. You can do anything with it you wish, but please don't ask us to spend any more time on it!

To download the sample program go to the DDE download area.

MFC DDE class libraries for C++

An MFC class including source code is available for download at **http://www.flashcast.com/dde32**

MSDN also has a project which wraps up DDEML in a class module. The project can be downloaded from **http://msdn.microsoft.com/library/techart/msdn_ddeplus.htm**

Question and topic list DDE Software downloads Home page

The missing data and unreliability problems

There are several ways in which DDE can fail to work properly.

- The server or client is not following the correct protocol.
- Failure to handle errors when using synchronous transactions.
- Using the fAckReq flag when establishing hot and warm links.
- The receiver does not get the messages sent to it.
- Failure to understand the limitations of Visual Basic DDE.

The server or client is not following the correct protocol

Because DDE is only a documented protocol for regular Windows messages, it cannot be policed by Windows. If both a server and client written as a pair disobey the protocol, they may work together all the same, but fail when used with a component which is programmed correctly. For example

- A server which asks for the receipt of messages to be acknowledged may continue to send data regardless of whether the acknowledgement has been sent.
- The client or server may ignore the clipboard format sent or requested.
- Messages may be sent out of sequence, such as data being sent on an advise loop before the advise request is acknowledged.

Failure to handle errors when using synchronous transactions

DDE is an asynchronous process, but is made to appear synchronous when the message sender chooses to suspend execution whilst waiting for a

reply. If the timeout period specified is not long enough, the transaction will appear to have failed. Further problems may arise when the message from the server finally arrives. A much better way is to work asynchronously, a callback routine is entered when the partner application replies.

Using the fAckReq flag when establishing hot and warm links

In one place the DDE documentation states that the fAckReq flag "Instructs the server to wait until the client acknowledges that it received the previous data item before sending the next data item. This flag prevents a fast server from sending data faster than the client can process it."

This is misleading. What is meant by "the next data item" is actually "the next data item after the ack is received". It does NOT mean "the next data item after the last one which was sent to you". The effect of the fAckReq flag is explained differently in another part of the MSDN literature.

"It is possible for a server to send updates faster than a client can process the new data. The speed of updates can be a problem for a client that must perform lengthy processing operations on the data. In this case, the client should specify the XTYPF_ACKREQ flag when it requests an advise loop. This flag causes the server to wait for the client to acknowledge that it has received and processed a data item before the server sends the next data item. Advise loops that are established with the XTYPF_ACKREQ flag are more robust with fast servers but may occasionally miss updates. Advise loops established without the XTYPF_ACKREQ flag are guaranteed not to miss updates as long as the client keeps up with the server."

The receiver does not get the messages sent to it

This happens if the message queue fills up under Windows 3.1 and 9x. If the client does not keep up with the server the message queue will eventually fill up. Messages are then lost because PostMessage fails. The only answer is to stop the server manually when (and if) prompted by Windows. The user interfaces will probably not respond. If the server continues attempting to post messages the system is likely to crash. The fAckReq flag was introduced to address this problem.

Under Windows NT it does not happen, the queue continues to grow. If the client catches up, the queue starts reducing again. If not, the task scheduling is changed to reduce it. The user interfaces will probably continue to respond, although Task Manager may report that the application is not responding. We have found Windows NT to be bullet-proof, you can safely choose not to use the fAckReq flag.

Failure to understand the limitations of Visual Basic DDE

This subject has its own section, please go to DDE and COM in Visual Basic

Question and topic list DDE Software downloads Home page

How can I improve the performance of my DDEML client?

In summary, by reducing the number of Windows DDE messages to the minimum and processing them as soon as possible.

First and foremost, if you are using synchronous calls, rewrite the client to use asynchronous transactions. When you make a synchronous request, the client process stops until either a response arrives, or the timeout expires. You will get much better performance if the client is completely callback driven.

In outline, before making a request for a piece of data, advise loop etc., see if there is already an outstanding request for it. If so, do not make another request but wait for the first request to be answered. You could cancel the transaction and start another, but that will double the number of Windows messages. If you need multiple items of data and use synchronous requests, each must wait for the previous one to succeed or time out. If you use asynchronous requests you can ask for them all at once, Windows will keep the messages in the queue and send them to the server when it is free. As each data item arrives your callback routine is invoked. The data can come in any order depending on the server, your client must be programmed to allow for this.

Secondly, use Advise loops instead of Request loops. A Request loop uses 3 Windows messages when the data changes (Notify, Request, Data), instead of only 1 with an Advise loop (Data).

Question and topic list DDE Software downloads Home page

Applications hanging because of DDE

There are two ways in which attempting to establish a DDE conversation can cause applications to stop responding.

A top level window does not have a message loop

This only happens in 32-bit Windows, and only DDE clients are affected. To make a connection, an application broadcasts a message to all top level windows. If the window is not on the same thread, the message is actually posted to the recipients queue and the caller is blocked. If the recipient has no message queue the calling thread is permanently blocked. Microsoft has acknowledged that this is a bug. Further details are in the Knowledgebase article Q136218 BUG: DdeConnect Never Returns.

Messages are lost because the wrong server accepts a connection

If a DDE server wrongly accepts a connection which ought to be accepted by another application it has a doubly bad effect. The client will probably lock up because the conversation is with the wrong non-responding server and the correct server never gets the connection. The rouge server will cause a problem or not apparently at random, because the order in which the windows receive the WM_DDE_INITIATE message is indeterminate. Any activity which depends upon DDE could be affected.

It is reported that Outlook causes problems with other applications trying to use DDE. We don't use Outlook ourselves, so we have not been able to test whether the cause is either of those mentioned here. Please e-mail ddefaq@rhaminisys.com if you know the answer.

Question and topic list DDE Software downloads Home page

Causes of slow DDE communication

In some quarters DDE has a reputation for being slow. This can be true if it is not programmed properly. Apart from general inefficiency of coding, the following are specific causes of slowness:

- Establishing a connection between the server and the client may be time consuming. Windows broadcasts a message to all top level windows, a connection is made when a server responds. A system which continually breaks and re-makes connections is going to be slow. Watch out for frequent calls of DDeInitiate in VBA, for example.

- Treating DDE transactions as synchronous, with the client waiting for the server to respond before proceeding. This is poor practice, DDE is especially suitable for cases in which the server may take a long time to acquire the requested data. Visual Basic uses synchronous operations for requests.

The speed of properly programmed DDE is similar to that of COM. For more details see COM and DDE data transfer speeds compared.

Question and topic list DDE Software downloads Home page

What are the 16/32-bit and Windows version issues?

As far as server and client applications are concerned, DDE is completely "bit blind". Neither can tell whether the other partner is 16-bit or 32-bit. There is however an effect on performance, DDE monitoring and the effect of programming errors.

Windows NT can handle a long message queue, other versions cannot

The Windows operating system will queue DDE messages if the server is delivering data faster than the client can use it. There is a discussion in the topic The missing data and unreliability problems.

Under Windows 3x and 9x 16-bit programs share the same address space and cooperate.

Usually, if both server and client are 16-bit the message queue does not grow indefinitely, because the operating system allows each to run in turn. Having a server which can deliver data faster than the client can use it is less likely to be a problem than with 32-bit programs.

32-bit programs are truly multi-tasking

A 32-bit server which puts messages on the queue faster than the clients can process them causes the queue to grow. Windows 9x runs out of resources and the server has to be shut down or Windows crashes (it may crash anyway). NT is completely stable, it is intelligent enough to give the client more of the processor time, allowing it to keep up with the server.

Whilst the queue is filling, the user interface of Windows 9x is usually unresponsive, in contrast to Windows NT which allows user input to be processed.

The 32-bit DDEML library is thread specific

The call to DdeInitialise and all subsequent DDEML API calls must be made in the same thread. The callback routine runs on the same thread. If you want more than one thread to use DDE, then each thread must have its own instance of the library, created by calling DdeInitialise with the pointer to the instance handle pointing at a zero hInstance.

DDE is asynchronous by nature, there is no need to use threads to allow a main thread to proceed whilst DDE operations are being carried out. The answer is to call DdeClientTransaction with a timeout of TIMEOUT_ASYNC, such calls return immediately.

Windows 9x and 3x use the 16-bit DDEML library, Windows NT has a native 32-bit implementation in User32.dll.

One result of this is that programming faults have different effects, for details see Why does my program run differently on Windows 9x and NT?.

Under Windows 9x 32-bit DDE using DDEML is slightly slower because of thunking down to 16 bits. Under Windows NT DDE between 16-bit programs is markedly slower than between 32-bit programs, in round figures about 2/3 of the speed (caution: your mileage may vary widely).

Applications which monitor DDE activity

The location of the DDEML API is important to DDE monitoring applications. DDEspy from Microsoft and our Dynamic Data Studio use the DDEML APPCLASS_MONITOR facility. Message hooks can read the memory handles of the data attached to a WM_DDE_xxx message, but for 32-bit applications the handles have no meaning in other process spaces. The 32-bit version of the DDEML API extracts and passes some data to a monitoring application, but only the first 28 bytes. There is no way to obtain more than this.

> ***Monitoring under Windows 9x*** 32-bit programs thunk down to 16-bit DDEML.DLL from User32.dll. As a consequence, all DDE transactions to and from 16 and 32-bit programs can be monitored by a 16-bit application. A 32-bit application cannot monitor anything.

Monitoring under Windows NT 4 16-bit programs call 16-bit DDEML.DLL and the Windows messages go to and from 32-bit programs via WOWEXEC. A 16-bit monitoring application uses 16-bit DDEML.DLL and can monitor only messages to and from 16-bit programs. A 32-bit application uses the native User32.dll interface and monitors messages to and from 32-bit applications.

Monitoring under Windows 2000 (NT 5) Unlike NT 4, a 32-bit monitor sees DDE messages going to and from 16-bit and 32-bit applications, although it only sees 32-bit DDEML activity. A 16-bit monitoring application can monitor only messages to and from 16-bit programs, as under NT 4.

Question and topic list DDE Software downloads Home page

Why does my program run differently on Windows 9x and NT?

The differences are due to the fact that Windows 9x uses the 16-bit DDEML library, whereas Windows NT has a native 32-bit implementation in User32.dll. Applications programmed exactly as specified in the documentation work the same way on both systems. If you do not control the source code and compilation, no remedy is available for the problems.

Some transactions do not complete on Windows NT

There is a sign extension 'feature' in both VB5 and VB6 (and probably other compilers). The constant &H8000 is compiled as &HFFFF8000. In VB it is not possible to write &H00008000, the leading zeros are removed. Some of the constants used in the DDEML API have the &H8000 bit set. Under Windows 9x User32.dll thunks down to 16-bit DDEML, which truncates the operation codes to 16 bits. But under Windows NT the DDEML API is native 32-bit code in User32.dll, and it does not remove the top 16 bits.

In Visual Basic there is a simple work-around, write the constant in decimal, for example write &H8000 as 32768. Writing &H8000& should also work. You **cannot** overcome this problem by making a 16-bit integer constant, such as
Public Const Foo As Integer = &H8000

because sign propagation on expansion to 32-bits happens also at run time.

The transmitted data (in either direction) is different

When you allocate a block of memory for DDE transfer it allocates a rounded size of block with padding at the end, and all of it is transmitted. Therefore, you must know how long your data is. Windows 9x and Windows NT appear to use a different granularity.

A client receives data under NT but not Windows 9x

If the server sends back data with a different clipboard format to the one asked for, DDEML.DLL (Win 9x) knows that the server has not sent what it was asked for and does not invoke the user's callback. It seems that user32.dll does not care and the callback is invoked.

Non-string data with Execute commands may not be passed correctly.

DDE works on the assumption that execute commands are strings. You may have difficulty and find differences between NT and 9x in passing data in other formats. In particular, NT may truncate data which has a zero second byte.

See also Windows NT NetDDE security and other issues.

Question and topic list DDE Software downloads Home page

Windows NT NetDDE security and other issues

The comments here apply to Windows 2000, there are additional considerations discussed in Windows 2000 NetDDE issues.

Net DDE shares must be marked "trusted" by an administrator

Under NT DDE shares must be "trusted" for a remote client to connect. This is managed by DDESHARE.EXE. However, only an administrator for the logged on workgroup or domain can trust a share. It is no good being an administrator for another domain. Confusingly, the dialog can be completed by a normal user and the share appears to have been trusted, but remote

clients cannot connect.

The client must give a password when connecting to a remote NT server

NT security gives password protection for access to remote machines. There is a bug in Windows NT, a password is always required when a client connects, even if one should not be required. The password is obtained from a dialog box. The client appears to connect immediately, because the DDE conversation is with Netdde.exe on the local machine. We have not found a workaround for this, the only hope appears to be programmatic completion of the dialog.

The number of NetDDE sessions allowed under Windows NT may be limited

By default, NETDDE applications are limited to 16 nodes. When the application attempts the 17th connection, it will fail and post an error to the event log. To fix this problem find the registry key

HKEY_LOCAL_MACHINE\Software\Microsoft\NetDDE\Parameters\NetBIOS

Edit or Add a REG_DWORD value named MaxSessions. Set it to a data value between 0 and 254 (decimal). If this value is missing or set to zero, the default of 16 is used.

Microsoft Knowledgebase articles

Q114089 Using Windows NT NetDDE Share Manager
Q140551 NetDDE Requires Username/Password When Connecting to WinNT
Q105195 NetLogon Service Necessary for NetDDE Connections in a Domain
Q142625 NETBIOS Defaults to 16 Sessions on Windows NT
Q164882 Practical Recommendations for Securing Internet-Connected Windows NT Systems
Q172450 NetDDE Connections Fail When "Logon to" Restrictions Apply

Question and topic list DDE Software downloads Home page

Windows 2000 issues

All the comments in Windows NT NetDDE security and other issues apply, and there are additional considerations.

NetDDE and Windows 2000

On a new Windows 2000 installation, NetDDE is disabled for all users. A client trying to connect gets a dialog box showing Incorrect Password no matter what is entered. You do not have the right to establish a NetDDE session because the account on the server has not allowed "Access This Computer from the Network" rights. To resolve the problem, prepare your mouse fingers for action and go to Settings/Control Panel/Administrative Tools/Local Security Policy/Local Policies/User rights assignment /Right click "Access this computer from the network"/Security. Then make sure the relevant user, domain or group is enabled. See these articles:

Default Access Control Settings in Windows 2000
Q256575 ClipBook Viewer Prompts for Network Access Password
Q257346 "Access This Computer from the Network" User Right Causes Tools Not to Work

Bugs in Windows 2000 DDE

- Connecting to an instance specific service does not work, an internal system error is reported. If more than one instance of an application is running you cannot connect to a specific one, unless the server application itself distinguishes between instances on the basis of the topics supported. For example, Excel supports topics based on the workbook and sheet name.
- There is a bug which may prevent a NetDDE connection from working. Details are in the Knowledgebase article Q272485 NetDDE Client Is Unable to Connect to NetDDE Server on First Attempt. Microsoft claim that this bug is fixed in Service Pack 2.
- The article Q181946 HOWTO: Create a NetDDE Client and Server in Visual Basic, contains a workaround for a bug in DDESHARE. When you trust the share, you must select "Initiate to Application Enable" and click the "Set" button before clicking OK on the two dialogs.
- It is necessary to set the DSDM service to Manual start. The NetDDE service may be Automatic or Manual, it starts DSDM itself. It was suggested by Microsoft to a reader of this page that allowing DSDM to start automatically can cause NetDDE not to work, and we have confirmed this.

Question and topic list DDE Software downloads Home page

DDE in NT Services

DDE needs a message pump for it to work. However, by default NT services don't seem to have one. Fortunately, the remedy is simple. When you install the service, you have to allow interaction with the desktop, even though it's not going to. The remarks in the following section concerning VB may apply to services written in any language.

Desktop interacting Services written in Visual Basic

Starting the service manually when a user is logged on is without problems. However, if the service is started automatically and it loads a form containing visible controls, it crashes with the system error 800706B5 (RPC unknown interface). The form does not have to be shown for the error to occur, loading it is enough. So, if you have a form for debugging or when the program is running as a regular application, do not load it when running as a service. Invisible controls such as our **DDClient** should be put on a separate form which is loaded but not made visible.

There is a further problem when a user attempts to log off. It is known to occur in Services using our DDClient control on an invisible form. We have not been able to circumvent it by using CreateObject() on the control instead. Windows attempts to unload the form containing the control. If the form is allowed to unload, the Control is removed. DDE stops and does not start again when a user logs on. If the Service prevents the form unloading in the QueryUnload event the user cannot log off.

The reason for putting DDE into a service in the first place is probably to allow automatic startup when the machine is switched on. It may not be a hardship to prevent the first user from logging off in these circumstances.

If you use DDEML, the default security arrangement is that the server is allowed to impersonate the client for DDE. For example, if you have a client running in the LocalService account, the server can run in any account.

Please e-mail ddefaq@rhaminisys.com if you know what the security situation is with raw DDE, or have any further insights into this topic.

Question and topic list DDE Software downloads Home page

Working with Unicode

This information is copied from MSDN. Note the implication the Execute transaction expects data to be a string. Attempting to pass data in other formats can cause problems.

"Applications that register a DDE window class that is Unicode must Unicode execute strings when communicating with other Unicode DDE windows. They must ASCII (OEM codepage) execute strings when communicating with non-Unicode DDE windows. Any window's state can be determined by calling IsWindowUnicode. This convention places burden of backward compatibility on Unicode DDE applications."

"Unicode compatibility is only a problem for raw DDE applications because DDEML automatically translates execute strings. DDEML bases its translation decision on how DDEML was called either by DdeInitiateA or DdeInitiateW when application first initialised conversation."

Question and topic list DDE Software downloads Home page

The "following System files have been replaced..." error message

If you get the "following System files have been replaced..." error message for DDEML.DLL every time you boot machine, answer is to delete DDEML.DLL related entry from Registry at

```
HKEY_LOCAL_MACHINE
   \System
       \CurrentControlSet
           \Control
               \SessionManager
                   \CheckVerDLLs
```

Question and topic list DDE Software downloads Home page

How to get further information

The primary sources of information are MSDN (Microsoft Developers Network) library and Microsoft Knowledgebase. Both are available on MSDN CDROMs and free on Internet.

Where to look in MSDN on CDROM or
http://msdn.microsoft.com/library/default.asp

Raw DDE using WM_DDE_xxx messages	Windows development/Windows Base Services/Interprocess communication/SDK Documentation/Interprocess communications/Dynamic Data Exchange
Using DDEML	Windows development/Windows Base Services/Interprocess communication/SDK Documentation/Interprocess communications/Dynamic Data Exchange Management Library
Network DDE	Windows development/Windows Base Services/Interprocess communication/SDK Documentation/Interprocess communications/Network Dynamic Data Exchange
Shell DDE interface	User Interface Design and Development/Windows Shell/SDK Documentation/Shell Programmers Guide/Shell Legacy Information/Shell Dynamic Data Exchange interface
Technical articles with sample code	Windows development/Windows Base Services/Dynamic Data Exchange/Technical Articles

Knowledgebase articles have an identifier like Q123456. You can obtain them on Internet in two ways-

1. Send an e-mail to **mshelp@microsoft.com**, with identifier as subject. You can also a space separated list of identifiers. Body text is ignored. To get an index of available articles subject is **index**.

2. View the article in your web browser. There is a generic form of URL

http://support.microsoft.com/support/kb/articles/q123/4/56.asp
If article number has only 5 digits, last part of URL is like **/q12/3/45.asp**.

There is also a good deal of information in help files that come with Microsoft's C/C++ compilers, especially API details.

Newsgroups are another good source of information.

Question and topic list DDE Software downloads Home page

Question and topic list DDE Software downloads

Back to the RHA (Minisystems) Ltd home page
http://www.rhaminisys.com