Dynamic Data Exchange - DDE

This document covers the following topics:

Concepts

DDE is a protocol defined by Microsoft Corp. to enable different applications to exchange data. This means that, for example, an application written in Natural may exchange data with a spreadsheet, because they are both able to process the DDE protocol. An application that processes the DDE protocol communicates with another DDE application via standardized messages. One of the applications is defined as the client, the other as the server. Client and server are holding a DDE conversation.

Anmerkung:
For an overview of DDE concepts and terminology, see your Microsoft Windows documentation.

Data in a DDE conversation is identified by a three-level hierarchy:

- service,
- topic,
- item.

A DDE conversation is established whenever a client requests a *service* from a DDE server. A DDE server offers one or more *services* to all active applications.

For each service, a DDE server may offer any number of *topics*. The DDE client then requests a conversation on a *topic* of a *service*.

In a conversation on a *topic* of a *service*, the DDE client and the DDE server uniquely identify data to be exchanged by an *item* name.

A DDE server may support a number of services, which in turn may consist of a number of topics, which themselves may contain a number of items.

With Natural, you can develop both DDE client applications as well as DDE server applications. You may, for example, write a Natural DDE client application that requests data from a spreadsheet acting as a DDE server, or you may write a Natural DDE server application that supplies a word processor (DDE client) with data.

To develop DDE client and DDE server applications, the following functionality is provided:

- A number of NGU-prefixed subprograms in library SYSTEM; these send messages and data as defined in the parameter data area NGULDDE1
- a parameter data area (NGULDDE1) which describes the parameters used by the subprograms in a DDE conversation (the DDE-VIEW);
- a DDE-Client event and a DDE-Server event which handle DDE messages.

You develop a DDE server application by reacting to the DDE-Server event and by using the NGU-SERVER-prefixed subprograms from library SYSTEM to register services and topics and to send messages and data to the DDE client application.

You develop a DDE client application by reacting to the DDE-Client event and by using the NGU-CLIENT-prefixed subprograms from library SYSTEM to initiate conversations and send requests and other DDE commands to DDE server applications.

You always have to include the parameter data area NGULDDE1 and the local data area NGULFCT1 in your client or server dialog. (You need NGULFCT1 in order to use the NGU-prefixed subprograms in library SYSTEM).

Developing a DDE Server Application

The following topics are covered below:

- Registering/Unregistering Services and Topics
- Getting Data From The Client
- Sending Data To The Client
- Terminating DDE Server Operation

Registering/Unregistering Services and Topics

Before a DDE server application can be addressed by a DDE client application, it must register its service names and all supported topics for the services. You use subprogram NGU-SERVER-REGISTER to do this for each service/topic the DDE server supports. Registering will usually be handled in the "after open" event of the base dialog.

When registering a service/topic for the first time, you will need to supply Natural with the dialog-ID of the dialog that will function as the server and that will therefore receive all DDE messages from clients. This is done by setting the `DDE-VIEW.CONV-ID` to the respective dialog-ID and also by setting `DDE-VIEW.MESSAGE` to the string "DLGID".

Note that at a later time you are able to add more topics to a service or even entirely new services. You can also make a topic unavailable by using subprogram NGU-SERVER-UNREGISTER.

Getting Data From The Client

After successful registration, it is possible that the DDE server application receives DDE messages from a DDE client application which is establishing a conversation on a registered topic of a service.

Such messages for a DDE server are received in the DDE-Server event of the dialog. At the beginning of the event-handler section, it is necessary to fill the DDE-VIEW with the client's message data. This is done by using subprogram NGU-SERVER-GET-DATA. After reading the data, it will be necessary to act based on the client message received. The possible messages and their meaning are explained in the description of subprogram NGU-SERVER-GET-DATA.

Sending Data To The Client

In many cases, the client message ultimately requires the server to send data to the client. This is achieved by using the subprogram NGU-SERVER-DATA.

Terminating DDE Server Operation

Whenever DDE server operation is supposed to terminate, you use the subprogram NGU-SERVER-STOP. It unregisters all services and terminates all active conversations. You terminate the server application with the `CLOSE DIALOG` statement.

Developing a DDE Client Application

The following topics are covered below:

- Connecting With The DDE Server Application
- Using The Services of a DDE Server Application
- Receiving Data From The DDE Server Application
- Disconnecting From The DDE Server Application
- Terminating DDE Client Operation

Connecting With The DDE Server Application

In order to establish a conversation with a DDE server application, a DDE client application must call the subprogram NGU-CLIENT-CONNECT with the service and topic name of the server it wants to connect. In order to receive the appropriate DDE events from a server, it is necessary to set the `DDE-VIEW.CONV-ID` to the client's dialog-ID and also to set `DDE-VIEW.MESSAGE` to the string "DLGID". The call will return a unique conversation ID in `DDE-VIEW.CONV-ID`. This value must be set appropriately in all further communication with the server.

Using The Services of a DDE Server Application

The client has several options to use the services of a server once a conversation has been established. It can

- request data on a specific item (using NGU-CLIENT-REQUEST),
- send data to the server (using NGU-CLIENT-POKE),
- ask the server to execute a command (using NGU-CLIENT-EXECUTE), or

- establish a warm or hot link to the server (using NGU-CLIENT-ADVISE-HOT, NGU-CLIENT-ADVISE-WARM and NGU-CLIENT-ADVISE-TERM).

Receiving Data From The DDE Server Application

The DDE client will receive data or other messages from the DDE server via the client dialog's DDE-Client event. Whenever a server has sent a message, this event occurs. The message contents must first be retrieved using NGU-CLIENT-GET-DATA. This will fill the DDE-VIEW structure appropriately. The client must then determine which message (DDE-VIEW.MESSAGE) has arrived and react appropriately. The possible messages are listed in the description of subprogram NGU-CLIENT-GET-DATA.

Disconnecting From The DDE Server Application

Whenever the client determines that the conversation is no longer needed, a call to NGU-CLIENT-DISCONNECT must be issued to inform the server that the conversation is to be terminated.

Terminating DDE Client Operation

Whenever the client application terminates or wants to stop using DDE, it needs to call NGU-CLIENT-STOP. This informs Natural to close all active conversations of the client and shut down DDE operation for the application.

Return Codes

Possible return codes are described in this section.

Anmerkung:
Each error-code description is not necessarily comprehensive. In these cases, the description is marked with an asterisk (*).

Code	Meaning
-1	You have specified an incorrect command or command parameter. Ensure that your DDE data area is of the correct type and that the command is correct.
0	The function was processed correctly.
1	This value is returned when an application has attempted to initialize with the DDEML library more than once. Check the logic of your program. Also ensure that the DDEML was exited correctly during the last run of the program.
2	This value may be returned from the server-initialize function if you have run the program before and not exited the DDEML correctly. It is also returned by a call-back function, whenever the requested service failed.
	An error occurred in the underlying layer. *
3	The conversation ID referenced does not represent an active conversation. Check if you have specified a correct service name.
4	The application could not initialize with the DDE library as the maximum number of instances are connected.

Code	Meaning
5	The DDEML communication has not been initialized. You must initialize with the DDEML before any DDE activity can take place.
6	Memory allocation problems encountered. This error might occur if the queue of messages for either part in the conversation becomes too long. *
7	A service, topic or item name was longer than 255 characters. Check if your fields are correctly specified for DDE-VIEW and make sure that you are not attempting to place a string longer than 255 characters in any one of the above variables.
8	An error occurred in the DDE library. Contact Software AG Support. *
9	Parameters passed to this function were illegal. This can be returned by any function call. Check your parameters.
10	"Server Type Link" is supported but no call-back function for UNLINK is passed to the function `PIDsRegisterTopic`. *
11	An attempt was made to remove a topic for which at least one conversation is still active. This includes trying to unregister a topic for which a conversation still exists.
12	The service/topic referenced has not been registered with the function `PIDsRegisterTopic`.
13	No links were active for the DDE-VIEW.SERVICE when the NGU-Server-Data subprogram was used. Check your service name and use the DDE-SPY in the SDK Tool Kit to see what services are available.
14	The requested type of link is invalid.
15	The transaction ID is corrupted. Check the value of your transaction ID in your DDE view.
16	The client application requested a conversation and prior to that, no function was specified to send the data for the links.
17	An asynchronous transaction was requested, but the client application did not specify a function to send details of the completed transaction. Such a function must be specified when the conversation is initialized.
18	A synchronous transaction timeout expired. The amount of time taken for your transaction to complete was longer than the TIMEOUT value in your DDE-VIEW structure. Increase the TIMEOUT value or set it to "-1" for indefinite waiting.
19 - 24	For internal use only.