DDEApp LIBRARY USER GUIDE

Supported OS	Windows 32-bit	
Supported Platform	Borland C++ Builder 6	V1.3.12
	Borland Developer Studio 2006	V1.3.12
	Microsoft.NET (Framework 2.0)	V2.0.0

Introduction

DDEApp is an application communication framework based on DDE communication protocol in order to exchange information between applications.

This library was first designed to simplify and standardize DDE implementation for applications built on Borland C++ platforms. We have further extended the library to .NET framework to provide bridge between .NET applications and C++ applications.

Using DDEApp Server (Step-by-step instruction)

DDEApp Server is designed to "serve" DDEApp Client request. It can handle request from multiple clients but only one request can be processed each time.

Initialize

Call constructor to instantiate DDEApp Server. Only one server instance is allowed for each application.

Example:

```
[C++] MDDEApp_Server ddeserver = MDDEApp_Server::SingleInstance(this);
[C#]  DdeAppServer ddeServer = DdeAppServer.SingleInstance(this);
```

Assign Event

Assign event functions to get notified for DDEApp Server events. Refer to programmer's reference for list of events available.

Example:

```
[C++]
ddeserver->OnClientConect = DDEClientConnect;
void __fastcall TFMain::DDEClientConnect(TObject *Sender){ /*Code*/ }

[C#]
DdeServer.ClientConnected += new
EventHandler<DdeAppEventArgs>(DdeServer_ClientConnected);

void DdeServer_ClientConnected(object sender, DdeAppEventArgs e)
{ /*Code*/ }
```

Register Functions and Variables

DDEApp Server is ready for action after these two steps. It can now respond to DDEApp Client's request. All built in internal function are ready for action. However, DDEApp Server has yet to have any information on functions and variables from your application. Thus, we need to tell DDEApp Server what are the functions and variables that can be access by DDEApp Clients.

Call `RegisterFunction()` / `RegisterVariable()` to expose application's function and variable for client access. Name must be single word and unique. Function name are **case insensitive**.

Variable type <*Type*> are defined as DDEVARTYPE which currently support types integer (dvInt), double (dvDouble) and string (dvAnsistring [C++], dvString[C+]). Set <*ReadOnly*> flag to true for read only access for clients.

Example:
[C++] `ddeServer->RegisterFunction(<Function Name>, <Description>);`
[C#] `ddeServer.RegisterFunction(<Function Name>, <Description>);`

Example:
[C++] `ddeServer->RegisterVariable(<Function Name>, <Type>, <ReadOnly>);`
[C#] `ddeServer.RegisterVariable(<Function Name>, <Type>, <ReadOnly>);`

Event OnFunctionCall must be assigned to process registered function whereas OnVariableRead and OnVariableWrite must be assigned to process variable read / write operation.

Error Handling

Please note that all user interactive action and message box prompt should be switch off when application is operate in DDEApp Server mode as any message box will freeze the server application. Exception thrown during DDEApp operation will be translated to message and forward to client who initiate the operation.

Ready for Action

Finally, you may use `Online()` and `Offline()` to enable / disable DDEApp Server when required. No client message will be processed in Offline mode.
Your DDEApp Server is all sets and ready for action.

Using DDEApp Client (Step-by-step instruction)

DDEApp Client on the other hands is used for send command and request to Servers. Each client can be connected to many DDEApp Servers.

Initialize

Unlike DDEApp Server, there is no restriction on number of instance allowed for DDEApp Client. Theoretically, you may create as many clients as you wish. Each client can connect to multiple servers. DDEApp Client is created by calling its constructor.

Example:

```
[C++] MDDEApp_Client ddeClient = new MDDEApp_Client(<MainForm>);
[C#]  DdeAppClient ddeClient = new DdeAppClient ();
```

NOTE: For C++ version, Application Main Form must be forwarded to DDEApp Client's constructor in order for DDE to work properly.

Assign Event

Assign event functions to get notified for DDEApp Client events. Refer to programmer's reference for list of events available. Read *DDEApp Server -> Assign Event* for example. Events in DDEApp Client mainly are optional and for notification.

Connect to Server

A link must be established between DDEApp Server and DDEApp Client before any function call or variable read / write can take place.

Example:

```
[C++]
int tHandle;
ddeClient->Add(&tHandle, <Server>, <Auto Launch>);    //Register
int tStatus = ddeClient->Conenct(tHandle);            //Connect

[C#]
string serverName;
serverName = ddeClient.Add(<Server>, <Auto Launch>);  //Register
ddeClient.Connect(serverName);                        //Connect
```

Register server:

- *<Server>* = target server application (full path)
- *<Auto Launch>* = automatic start server application if not started.
- *serverName* = reference name returned for current added server. Use for subsequent operation.

Connect to Server:

- Establish connection between client and server.

- If Auto Launch is enabled, DDEApp Client will attempt to connect with DDEApp Server when first registered.
- [C++] Function connect return NULL if success.
- [C#] Exception throw if connection if not success.

Execute Function

To execute function in DDEApp Server, all we just need is to execute `FunctionCall()` from DDEApp Client.

Example:

```
//Execute function without wait for return.
// handle OnFunctionReturn to get return status.
[C++] ddeClient->FunctionCall(tHandle, <Command>);
[C#]  ddeClient.FunctionCall(serverName, <Command>);

//Execute function and wait for return.
// function must returned before timeout.
[C++] ddeClient->FunctionCall(tHandle, <Command>, &tStatus, &tReturn);
[C#]  ddeClient.FunctionCall(serverName, <Command>,
      FunctionResult result);
```

Variable Access

Value for registered variable in DDEApp Server can be read via the following function.

Example:

```
[C++] ddeClient->VariableRead(tHandle, <VarName>, <ReturnValue>);
[C#]  ddeClient.VariableRead(serverName, <VarName>,
      VariableResult <Result>);
```

Call `VariableWrite()` in order to write value to variable which has write access.

Example:

```
[C++] ddeClient->VariableWrite(tHandle, <VarName>, <NewValue>);
[C#]  ddeClient.VariableWrite(serverName, <VarName>, <NewValue>);
```

Help Screen

There is a built in help information in DDEApp Server which allow DDEApp Client to get information such as basic usage information, library version, registered functions and variables from server. These information can be access by execute "Help" function from DDEApp Client.

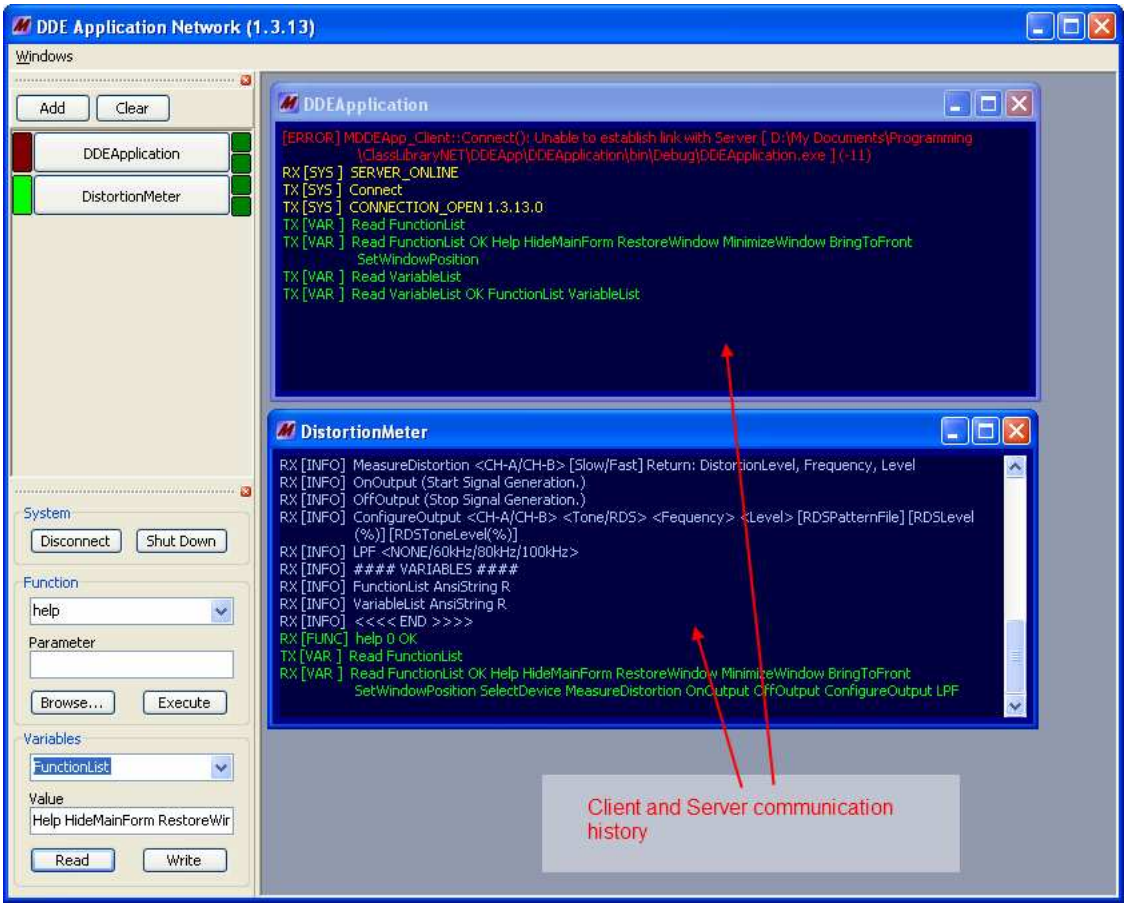
Internal Functions

DDEApp Server built in with some basic general functions such as: *Help, HideMainForm, RestoreWindow, MinimizeWindow, BringToFront* and *SetWindowPosition.* These functions name are internally reserved and may not be override. In other words, user is not allowed to use these names for their functions.

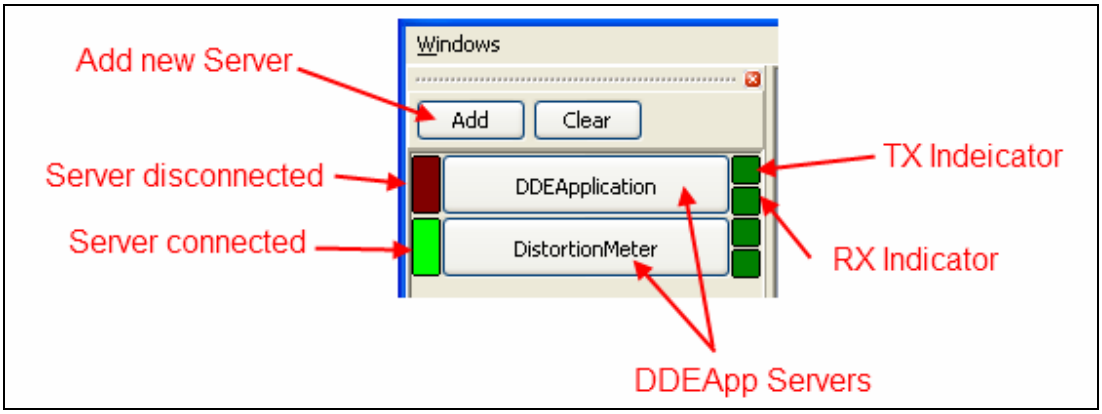
Using DDE Application Tool

About

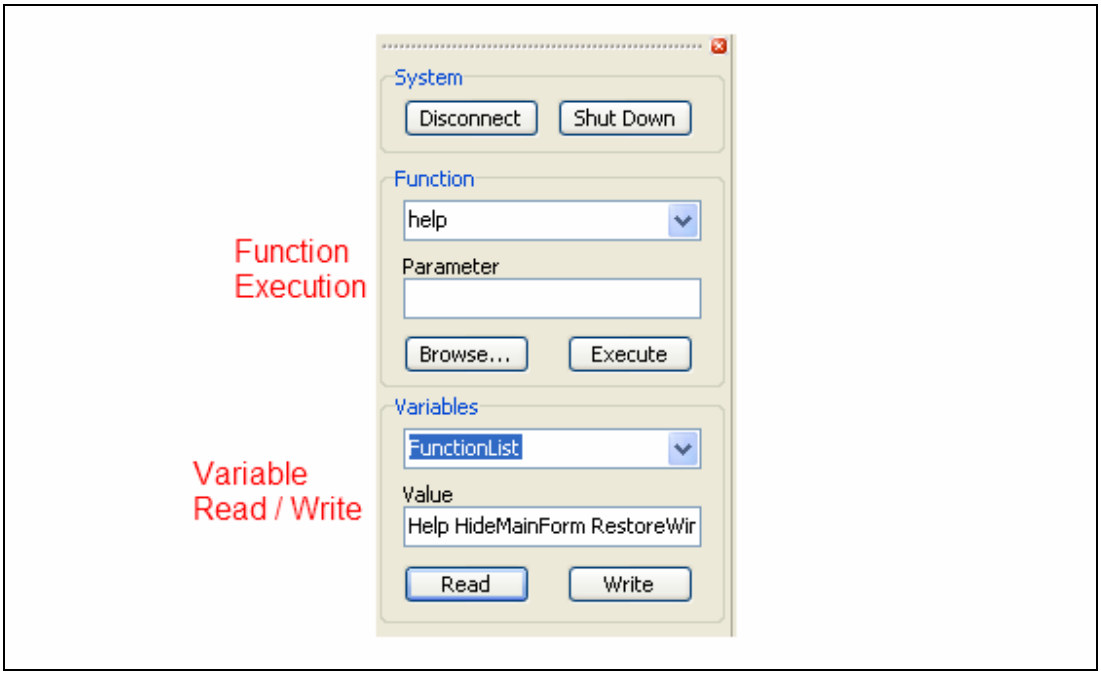
DDE Application is a tool to simulate DDEApp Client. This tool can be used to test DDEApp Server enabled applications.



DDE Application overview



Servers Manager



Function Execution and Variable Read Write