.zzf configuration file

Used by File Transfer to save settings to be used when connected to a remote system.

Details saved
General name
Initial directory
Anonymous login options
Auto connect
Listing type
Log details
Listing type details
Transfer options
Refresh options

.zzc configuration file

Used by File Transfer and terminal emulators to save the connection details of remote systems.

Details saved

Remote system name
User name
Protocol details
Password request

.zzt configuration file

Used by terminal emulators to save details to be used by a remote application.

Details saved

Remote application help sequence
Terminal emulator closure sequence
Information exchange details
Fonts
Keyboard mapping details
Connection details
Startup options
.zzs file name containing scripting details

.zze configuration file

Used by terminal emulators to save emulator properties.

Details saved
Color mapping
Number of lines in display and scrollback buffer
Cursor style
CR / LF mapping
Window update options

.zzk configuration file

Used to save keyboard mappings independently of a terminal emulator. This type of configuration file can be loaded and used by a terminal emulator.

Details saved

Key mappings
Keys assigned to escape sequences
Keys assigned to .zzs script files

.zzs configuration file

Used by terminal emulators to save details on scripting. This type of configuration file can be loaded and used by a terminal emulator.

Details saved

Connection and login information and commands to launch remote applications.

Used by the web browser and Organizer to save details on bookmarks. This type of configuration file can be loaded and used by the web browser and Organizer.

Details saved

Details on the contents of the bookmarks are saved. The Organizer also saves Briefcase and History details of the locations visited in this file.

> mvtchrefUsed by Mail to save messages. This type of configuration file can be loaded and used by Mail.

Details saved

Details on any unsent messages (from working off-line) or saved messages.

.zzr configuration file

Used by a terminal emulator, to save details on active URLs and other configured rules for the application currently in use.

Details saved

Details on `http://`, `ftp://`, `file:///`, `telnet://`, `news:`, `mailto:` and `gopher:` URLs that have been activated for the application in use.
Details on any other rules that have been configured.

DDE overview

Dynamic data exchange enables application interaction through a messaging system. Two Windows-based applications can communicate via a DDE conversation, sending messages to each other.

DDE applications can be built using DDE supported languages such as Visual Basic and Visual C++. Communication can also take place between existing DDE supported applications for example Microsoft Excel.

DDE conversation occurs between a *server* application which contains data that may be of use to other applications and a *client* application that initiates a conversation and requests data from the server. In MultiView 2000, the terminal emulator acts as the server and any application requesting data acts as the client.

Within MultiView 2000, DDE can be used to provide an alternative connection procedure for terminal emulators and for exchanging information for example a terminal emulator providing data to update a graphical chart in Excel.

Related Topics

A typical DDE conversation

1 The client application broadcasts an initiate message to begin a conversation. The message specifies the name of the application to communicate with and the topic. The server application that recognizes the application and topic name, responds to the client and the conversation begins. If more than one reply is received, the client has a choice of servers.

2 Once a DDE conversation is established, the client can send the following requests:

DDE_REQUEST request a specified data item from the server.

DDE_ADVISE request continuous status advice from the server for a specified data item.

DDE_POKE request that the server receive the specified data item.

DDE_EXECUTE requests that the server execute the specified command.

3 The server can send data to the client or execute a command sent by the client.

4 The client can end the conversation by sending a DDE_TERMINATE message.

Types of DDE conversation

There are three types of conversation links that can be used:

cold link
The client sends messages to the server requesting specific data. The server responds by sending the requested data. The client is not notified when data on the server changes.

hot link
The client initiates the conversation by requesting some data. Once the link for a data item is established, the server will notify the client whenever the state of this item changes by sending the updated item.

warm link
This link combines the features of the hot and cold links. Once a link is established, the server only notifies the client of a change in a data item but does not send the updated data. The client must request the data item if it is required.

DDE data format

To participate in a DDE conversation, both the client and server need to understand the format of the data. This is done with the application, the data topic and the data item:

Application
The server application or program name. This would usually be the name of the remote application.

Topic
A combination of the server application name and the full path of the .zzt file which contains the remote system login details. The topic is used to identify the window of the server application which is required by the client to carryout a DDE conversation.

Item
The main use of an item is to match a client request with a server response.

Two different types of item can be can be used:

1. JSBDDEStream - a circular buffer which can contain up to 4 standard screens of data (24 lines by 80 columns).

See connecting to a remote system for more details.

2. The item name contains screen coordinates of the server data item that is to be retrieved by the client.

There are two ways to specify the area:

Block mode: x, y coordinates, length and height on the screen.
e.g. X11Y5L5H3

Line mode: line mode type, x, y coordinates, length and height on the screen.
e.g. LX11Y5L5H3

Connecting to a remote system using DDE

DDE can be used to interact with MultiView 2000 by specifying an alternative way of connecting to a remote system via a terminal emulator. In such an example, control of the login procedure is passed from the terminal emulator to a DDE client application which will handle requesting of data from the communications server via the terminal emulator which is the server. The client will process data sent by the communications server and send data back to the communications server to facilitate a connection.

To enable a DDE connection, a new item called JSBDDEStream is used as the Item element in a DDE conversation. The JSBDDEStream link is kept in a 4K bytes circular buffer which can store the equivalent of up to 4 standard screens of data (24 lines by 80 columns) before the buffer overflows. If data is not read before the buffer overflows, the old data is lost.

For cold and warm DDE links, the current contents of the buffer is sent to the DDE client when it receives a request. The buffer pointers are then reset to allow another 4K bytes of data before overflow. Hot links send the data at the rate it arrives from the server.

Related Topics

Exchanging information using DDE

MultiView 2000 allows DDE to be used to exchange information between a terminal emulator or server and another DDE client application. For example, a specified field in a remote database could be identified as a DDE link. In Excel using Paste Link, a cell is created to include the data from the remote database field. If the remote field is updated, the related cell in Excel is automatically updated using a hot link.

DDE supported functions

Abort

Show(type)

Move(x, y, W, H)

Abort

Parameters

None

Comments

Quits the DDE program.

Visual Basic example

```
txtLink.LinkExecute "[Abort]"
```

Where `txtLink` is the text box which supports the DDE link between the Visual Basic application and the terminal emulator.

Show(type)

Parameter

type	One of the following flags can be passed in this function:
(0)	Hide the window and pass activation to another window.
(1)	Activate and display a window. If the window is minimized or maximized, it is restored to its original size and position (same as flag 9 below).
(2)	Activate a window and display it as an icon.
(3)	Activate a window and display it as a maximized window.
(4)	Display a window in its most recent size and position. The currently active window remains active.
(5)	Activate a window and display it in its current size and position.
(6)	Minimize the specified window and activate the top-level window in the system list.
(7)	Display a window as an icon. The currently active window remains active.
(8)	Display a window in its current state. The currently active window remains active.
(9)	Activate and display a window. If the window is minimized or maximized, it is restored to its original size and position (same as flag 1 above).

Comments

Shows the terminal emulator window.

Visual Basic example

```
txtLink.LinkExecute "[Show(1)]"
```

where `txtLink` is the text box which supports the DDE link between the Visual Basic application and the terminal emulator.

Move(x, y, W, H)

Parameters

x, y, W, H	x, y coordinates, Width and Height respectively.

Comments

Moves the terminal emulator window to the specified location.

Visual Basic example

```
txtLink.LinkExecute "[Move(1,1,400,200)]"
```

where `txtLink` is the text box which supports the DDE link between the Visual Basic application and the terminal emulator.

Example 1 DDE conversation

An example of connecting to a specified remote system and running an example "sysadmin" application.

All the files required to use this application are supplied on the installation media in \samples\dde.
The files included are:

```
apps.frm
frmsyste.frm
dde.bas
m2000dde.mak
m2000dde.exe
```

You can use these files with Visual Basic to change the lines indicated below to suit your environment and compile a working version.

It is recommended that the MultiView 2000 folder is included in your PATH.

1 Create a .zzt configuration file using a terminal emulator to specify the remote system to connect to. In the Startup tab of the Properties dialog box, click Auto Connect.

2 Open the `m2000dde.mak` project make file with Visual Basic. This will also open the associated .frm files.

3 In the `frmsyste.frm` file, make the amendments indicated below:

   In the `Sub cmdConnect_Click` procedure, specify the application and configuration file, for example;

   ```
   StartApp("JSBTERM","-s" & chr$(34) & "C:\Program Files\MultiView 2000\jsb.zzt" & chr$(34))
   ```

   In the `CreateLink` procedure specify the topic, for example;

   ```
   ctl.LinkTopic = "JSBTERM|","-s" & chr$(34) & "C:\Program Files\MultiView 2000\jsb.zzt" &
   chr$(34)
   ```

   References to chr$(34) (double quotes) indicates that long file names (longer than eight characters) are being used.

4 Compile the project and run `m2000dde.exe`.

Another DDE example.

Example 2 DDE conversation

Part of an example of creating a connection with a DDE conversation in Visual Basic is given below:

Prior to running the program, a .zzt configuration file containing the remote system login information will be required.

```
### Starting a TE session and creating a DDE link
### Start a terminal emulator using login details specified in the jsb.zzt configuration file
If StartApp("JSBTERM", "-s" & chr$(34) & C:\Program Files\MultiView 2000\jsb.zzt" & chr$(34)) Then
    Select Case CreateLink(txtLinkControl, LINK_NOTIFY)
    .
    .
EndIf
.
.
### Check the DDE stream for a string and send to TE
If InStr(1, sLoginString, "ogin:") > 0 Then
   lblllogin.Caption="Please Wait... Sending Login"
EndIf
txtLinkControl.Text=(txtUserId.Text & Chr$(13))
txtLinkControl.LinkPoke
.
.
### Showing a TE session.
txtLinkControl.LinkExecute"[Show(0)]"
.
.
### Function to start an application
Function StartApp (appname As String, appargs As String) As Integer
   On Error Resume Next
   StartApp = (Shell(appname & " " & appargs) > 31)

   If Err Then
      MsgBox "Couldn't start " & appname & " " & appargs
      StartApp = 0
   End If
End Function

### Function to create a DDE link
Function CreateLink(ctl As Control, LinkType As Integer)As Integer
   On Error Resume Next
   ctl.LinkMode=NONE
   ctl.LinkTopic="JSBTERM|","-s" & chr$(34) & "C:\Program Files\MultiView 2000\jsb.zzt" & chr$(34)
   ctl.LinkItem="JSBDDEStream"
   ctl.LinkMode=LINK_NOTIFY
   CreateLink=Err
End Function

### Procedure to close down a TE session
Sub cmdCancel_Click()
   On Error resume
   If txtLinkControl.LinkMode > 0 Then
    txtLinkControl.LinkExecute"[Abort]"
   EndIf
   txtLinkControl.LinkMode=NONE
   Unload frmSystemConnect]

End Sub
```

This example will link a Visual Basic client program to a remote system using a terminal emulator, with the login details specified in the .zzt configuration file.

Full working DDE example.

File Transfer Protocol

To transfer files using FTP, make sure that TCP/IP is correctly configured on your PC

A feature of an FTP server is to time out when a connected File Transfer session remains idle for some time.

To configure the TCP/IP port for FTP

1 In File Transfer, click the Remote menu and click Add.

2 With the FTP protocol selected, click Configure.

3 Use the up-down buttons to select a port number and enter the appropriate Account Name.

Tips

- Depending on how the remote system is set up, not all systems require an account name.
- An account can be used to specify file permissions for users.
- You can configure an existing remote system which is disconnected by using a right click on the system and clicking the Configure command.

To specify number of folders to cache

You can specify the number of folders to be cached to improve the speed of displaying folders. On return to a cached folder, its details will be read from memory rather than its physical location.

The number of folders to cache can be set in the Refresh tab of the Options dialog box.

OLE Automation overview

OLE automation server applications allow their OLE created objects to be exposed to other applications which can then manipulate them.

Interaction between the object and the automation client is provided by a group of predefined functions. This allows for deviation or additional functionality to that offered by the automation server. A set of macro scripts incorporating the functions can be written in any language that will support automation for example Visual Basic or Visual C++.

In MultiView 2000, OLE Automation can be used to provide an alternative connection procedure for terminal emulators.

Connecting to a remote system using OLE Automation

OLE Automation can be used to interact with MultiView 2000 by specifying an alternative way of connecting to a remote system via a terminal emulator. In such an example, control of the login procedure is passed from the terminal emulator to an OLE Automation application or *client* which will handle requesting of data from the communications server via the terminal emulator which is the *server*. The client will process data sent by the communications server and send data back to the communications server to facilitate a connection.

This method bypasses the default login procedure which may be useful for example when a different login front end is required. Another use would be to provide an invisible terminal emulator required to run a remote application which does not need to be apparent to the user.

Exchanging information using OLE Automation

MultiView 2000 allows OLE Automation to be used to exchange information between a terminal emulator or server and another OLE Automation application or client. For example, a specified field in a remote database could be identified using a client application which would then also be responsible for sending and retrieving data from the server application.

OLE Automation supported functions

VT_BOOL Connect(VTS_NONE)

VT_BOOL Disconnect(VTS_NONE)

VT_BOOL GetRectChars(VTS_I2, VTS_I2, VTS_I2, VTS_I2)

VT_BOOL MoveWindow(VTS_I2, VTS_I2, VTS_I2, VTS_I2)

VT_BOOL OpenFile(VTS_BSTR)

VT_BOOL Quit(VTS_NONE)

VT_BOOL ResetStreamMode(VTS_NONE)

VT_BOOL RetrieveData(VTS_NONE)

VT_BOOL SendData(VTS_NONE)

VT_BOOL SetStreamMode(VTS_NONE)

VT_BOOL ShowWindow(VTS_I2)

PROPERTY szReadBuffer

PROPERTY szWriteBuffer

VT_BOOL OpenFile(VTS_BSTR)

Parameter

VTS_BSTR	Specifies a terminal emulator-based configuration file (`.zze` or `.zzc` or `.zzt` or `.zzk`).

Comments

Opens the terminal emulator with the specified configuration file. To open several configuration files for a single emulator, call this function several times with the appropriate parameter.

Note a configuration file must be specified.

VT_BOOL Connect(VTS_NONE)

Parameters

None

Comments

Makes a connection to the remote system specified in the `.zzc` configuration file.

VT_BOOL Disconnect(VTS_NONE)

Parameters

None

Comments

Disconnects from the remote system specified in the `.zzc` configuration file.

VT_BOOL ShowWindow(VTS_I2)

Parameter

VTS_I2	One of the following flags can be passed in this function:
(0)	Hide the window and pass activation to another window.
(1)	Activate and display a window. If the window is minimized or maximized, it is restored to its original size and position (same as flag 9 below).
(2)	Activate a window and display it as an icon.
(3)	Activate a window and display it as a maximized window.
(4)	Display a window in its most recent size and position. The currently active window remains active.
(5)	Activate a window and display it in its current size and position.
(6)	Minimize the specified window and activate the top-level window in the system list.
(7)	Display a window as an icon. The currently active window remains active.
(8)	Display a window in its current state. The currently active window remains active.
(9)	Activate and display a window. If the window is minimized or maximized, it is restored to its original size and position (same as flag 1 above).

Comments

Shows the terminal emulator window in one of the above states.

VT_BOOL MoveWindow(VTS_I2, VTS_I2, VTS_I2, VTS_I2)

Parameters

VTS_I2	x, y coordinates, width and height respectively.

Comments

Moves the terminal emulator window to the specified location.

VT_BOOL Quit(VTS_NONE)

Parameters

None

Comments

Quits the client application.

VT_BOOL SetStreamMode(VTS_NONE)

Parameters

None

Comments

Sets the server to stream mode. In this mode, the server will not read any connection data unless requested by the client application. The client application will then process the requested data which is passed on by the terminal emulator. This function allows for the substitution of an external login procedure if required.

VT_BOOL ResetStreamMode(VTS_NONE)

Parameters

None

Comments

Resets the server to normal processing mode. The server will read data when needed and process it accordingly.

VT_BOOL RetrieveData(VTS_NONE)

Parameters

None

Comments

The client application notifies the terminal emulator to read some data from the communications server into the property szReadBuffer.

VT_BOOL SendData(VTS_NONE)

Parameters

None

Comments

The client application notifies the terminal emulator to send the data that is stored within the property szWriteBuffer to the communications server.

VT_BOOL GetRectChars(VTS_I2, VTS_I2, VTS_I2, VTS_I2)

Parameters

VTS_I2	x, y coordinates, width and height respectively.

Comments

Retrieves the characters that are stored in the specified screen location from the server application.

PROPERTY szWriteBuffer
Stores the data to be sent to the communications server.

PROPERTY szReadBuffer
Stores the data that has been received from the communications server.

Creating OLE objects within an OLE application

The following objects can be created by the terminal emulator using an OLE application:

JSBTerm.Document - creates a terminal emulator object

JSBXfer.Document - creates a file transfer object

To disable the terminal emulator configuration options

1 Within the terminal emulator you need to deselect the toolbar views:

Click to clear the Toolbar option on the View menu.

Click to clear the Color Intensity Bar option on the View menu.

Close the terminal emulator.

2 Within the Windows Registry editor (`regedit`) you need to set the appropriate user level. You can do this for all users on this PC, or a single user. See below.

3 Remove the following components from the MultiView 2000 folder, normally:
`\Program Files\MultiView 2000`

Terminal Session Wizard:	`jsbtwiz.exe`
	`jsbtwizr.dll`
Rules Agent Editor:	`jsbruled.exe`

Disable a single user…

1 Locate the key:

`HKEY_CURRENT_USER\Sofware\JSB\MultiView 2000\Common\Emulators\`

2 Add a new DWORD:

- Name **User Level**
    - Data set to decimal, value 0-3. See table below.

Disable all users on this PC…

1 Locate the key:
`HKEY_USERS\.Default\Sofware\JSB\MultiView 2000\Common\Emulators\`

2 Add a new DWORD:

- Name **User Level**
- Data set to decimal, value 0-3. See table below.

User Level Table:

Value	**Meaning**
0	Full configuration available.
1	Configuration restricted.
2	Copy + Paste restricted.
3	Configuration AND Copy + Paste restricted.

Tips

- The User Level key must be set on all PCs that are to use terminal emulators with disabled configuration options.
- If the Toolbar is not removed, the user will still be able to connect to and disconnect from a selected remote system.
- The default value for the **User Level** key is 0, full configuration.

To embed a terminal emulator as an Active X document

The terminal emulator is an Active X document server which can be embedded in container applications that support Active X such as Microsoft Binder and Internet Explorer version 3.0 and above.

1 Within the Active X application container, click the File menu and click Open.

2 Enter or browse for the location of a `.zzt` configuration file that has previously been created.

3 Open the selected the file.

Tips

- A terminal emulator may also be loaded into a frame of a web page by including the URL of the `.zzt` configuration file within the html source of the web page.
- Relevant terminal emulator toolbar and menu options are incorporated in the application's interface.
- The terminal emulator can also be embedded as an object in OLE container applications.

Integrating the PC X server

The PC X server component is a cost option to MultiView 2000. Once installed, your Remote Command can now be configured as an X client. When used in conjunction with scripting, the Application button will now display a full X client command.

1 Click here to open the Properties dialog box. Select the Application tab.

2 In the Remote Command box, type the name of the X client to be executed.

3 Check Run as an X Client.

4 Click the X Advanced button.

5 In the X Client Advanced Details dialog, change the default details as required.

The X client is now configured for use when the Application token of the learn script procedure is used.

Tip

- This dialog box only contains X related configuration options if the PC X Server is already installed..
- This dialog box can be opened from the Application command on the Configure menu.

send

Defines a sequence to send to the terminal emulator.

send="*sequence*"

"*sequence*" is the send sequence and should be included within quotation marks (").

Example: send="$LOGIN" is a typical send command using the token $LOGIN.

Related Topics

receive

Defines a sequence to be received from the terminal emulator.

receive="*sequence*"

"*sequence*" is the sequence required from the terminal emulator and should be included within quotation marks (").

Example: receive="Password"

Is a typical receive command which could also include tokens.

Related Topics

delay

Defines a delay in seconds.

delay=*seconds*

seconds is the number of delay seconds.

Example: delay=1

Adds a 1 second delay in command sequence executions.

wait

Specifies the maximum period to wait for a string specified in the receive command.

Example: receive=*seconds*

seconds is the number of seconds

Example: receive="OK", wait=2

Specifies the terminal emulator to wait a maximum of 2 seconds to receive the string OK.

connect
Opens a connection between the specified terminal emulator and remote system using the selected protocol.

Supported script commands

A script file is made up of a set of command lines instructing the communications to connect a terminal emulator.

The following commands can be used in a script file:

connect
delay
receive
send
wait

Script tokens

Command lines in script files may include tokens listed below. Tokens always begin with a dollar ($) introducer and are typically used to reference a value previously entered or configured elsewhere in the terminal emulator.

$LOGIN	Defines the user name for logging in to the remote system. The login name may be preconfigured in the Connection tab of the Properties dialog or set in the Login To dialog box.
$PASSWORD	Defines the password associated with the $LOGIN name defined above and is taken from the Login To dialog box, if relevant, for the connection.
$TERM	Defines the terminal environment variable which is set in the Terminal tab of the Properties dialog box.
$APPLICATION	Specifies a remote application or command to execute on the remote system once a connection has been achieved. The remote command is set in the Application tab of the Properties dialog box.

Overview of script files

Script files can be used for automating login procedures and for running remote applications on connection to a remote system (they are independent of DDE script files). The `.zzs` configuration file used to save script commands can be created using the Remote menu script commands or the scripting toolbar within a terminal emulator window or by using a text editor.

These configuration files can be replayed at any time or on startup of a terminal emulator. Script files can also be mapped to a key in the Keyboard Mapping dialog box.