ZylAppCommunicator 1.29

ZylAppCommunicator is a Delphi / C++Builder component which allows you to communicate between one or more applications which are running on the same computer.
Use ZylAppCommunicator if you want to easily communicate between your applications.

The demo version is fully functional in Delphi and C++Builder IDE, but it displays a nag dialog (the licensed version will, of course, not have a nag dialog and will not be limited to the IDE). The package includes demo programs for Delphi and C++Builder and a help file with the description of the component.

Supported Operating Systems: Windows 2000/XP/Server2003/Vista/Server2008/7/8/Server2012/10/11
Available for: Delphi 12 Alexandria (Win32 & Win64), Delphi 11 Alexandria (Win32 & Win64), Delphi 10.4 Sydney (Win32 & Win64), Delphi 10.3 Rio (Win32 & Win64), Delphi 10.2 (Win32 & Win64), Delphi 10.1 (Win32 & Win64), Delphi 10 (Win32 & Win64), Delphi XE8 (Win32 & Win64), Delphi XE7 (Win32 & Win64), Delphi XE6 (Win32 & Win64), Delphi XE5 (Win32 & Win64), Delphi XE4 (Win32 & Win64), Delphi XE3 (Win32 & Win64), Delphi XE2 (Win32 & Win64), Delphi XE, Delphi 2010, Delphi 2009, Delphi 2007, Delphi 2006, Delphi 2005, Delphi 7, Delphi 6, Delphi 5, Delphi 4, C++Builder 12 Alexandria (Win32 & Win64), C++Builder 11 Alexandria (Win32 & Win64), C++Builder 10.4 Sydney (Win32 & Win64), C++Builder 10.3 Rio (Win32 & Win64), C++Builder 10.2 (Win32 & Win64), C++Builder 10.1 (Win32 & Win64), C++Builder 10 (Win32 & Win64), C++Builder XE8 (Win32 & Win64), C++Builder XE7, C++Builder XE6, C++Builder XE5, C++Builder XE4, C++Builder XE3, C++Builder XE2, C++Builder XE, C++Builder 2010, C++Builder 2009, C++Builder 2007, C++Builder 2006, C++Builder 6, Turbo Delphi, Turbo C++
Remarks:
- The Delphi 2006 version is fully compatible with Turbo Delphi
- The C++Builder 2006 version is fully compatible with Turbo C++
Limitations:
- Windows Vista/2008/7/8/2012/10/11 limitation: cannot be used for communication between applications with different isolation level.

Insatallation:
If you have a previous version of the component installed, you must remove it completely before installing this version. To remove a previous installation, proceed as follows:
-Start the IDE, open the packages page by selecting Component - Install Packages
-Select ZylAppCommunicatorPack package in the list and click the Remove button
-Open Tools - Environment Options - Library and remove the library path pointing to ZylAppCommunicator folder
-Close the IDE
-Browse to the folder where your bpl and dcp files are located (default is $(DELPHI)\Projects\Bpl for Delphi, $(BCB)\Projects\Bpl for C++ Builder). -Delete all of the files related to ZylAppCommunicator
-Delete or rename the top folder where ZylAppCommunicator is installed
-Start regedit (click Start - Run, type "regedit.exe" and hit Enter). Open the key HKEY_CURRENT_USER\Software\Borland\<compiler>\<version>\Palette and delete all name/value items in the list related to ZylAppCommunicator. (<compiler> is either "Delphi" or "C++Builder", <version> is the IDE version you have installed)

-Unzip the zip file and open the ZyAppComunicatorPack.dpk file in Delphi (ZylSerialPortPack.bpk file in C++Builder), compile and install it

and add to Tools/Environment Options/Library (in older Delphi/C++Builder menu) or Tools/Options/Delphi Options/Library/Library Path (in newer Delhi menu) or Tools/Options/C++ Options/Paths and Directories/Library Path & Include Path (in newer C++Builder menu) the path of the installation (where the ZyAppComunicator.dcu file is located). The component will be added to the "Zyl Soft" tab of the component palette. After you have the component on your component palette, you can drag and drop it to any form, where you can set its properties by the Object Inspector and you can write event handlers selecting the Events tab of the Object Inspector and double clicking the preferred event.
If you still have problems in C++Builder, running an application, which contains the component, then open the project and in C++Builder menu, Project/Options/Packages and uncheck "Build with runtime packages".

-It is indicated to use this component with "Stop on Delphi exception" option deactivated. You can do this from Delphi / C++Builder menu, Tools/Debugger Options/Language Exceptions/Stop on Delphi exceptions in older versions or Tools/Options/Debugger Options/Embarcadero Debuggers/Language Exceptions/Notify on language exceptions in newer versions, otherwise you will have a break at all the handled exceptions.

Types:

TReceiveEvent = procedure(Sender: TObject; SenderHandle: HWND; BufferType:Integer; Buffer: TStream) of object;
TZylAppCommunicator = class(TComponent)

Properties:

Enabled: Boolean - Enable / disable the communication.
Handle: HWND - Communicator identifier, unique for each communicator.
CommunicationPortID: AnsiString - CommunicationPortID must be unique for each communication protocol (implemented by you between two or more applications) to avoid communication interferences. When you register the component you will get a set of recommended values for this property to avoid interferences with other applications.
Version: Double - Version number of the component.

Methods:

constructor Create(AOwner: TComponent) - constructor
destructor Destroy() - destructor
procedure SendString(Target: HWND; Text: ShortString) - Sends string to a target communicator, identified by the "Target" parameter. "Text" parameter represents the string you want to send.
procedure SendString(Text: ShortString) - Sends string to all the communicators with the same "CommunicationPortID". "Text" parameter represents the string you want to send.
procedure SendBuffer(Target: HWND; var Buffer; const BufferSize: Integer; const BufferType: Integer) - Sends data to a target communicator, identified by the "Target" parameter. "Buffer" parameter represents the data you want to send and it can be of any type. "BufferSize" represents the size of the data in bytes you want to send. "BufferType" represents the type of data you want to send and it must be different from 0, because 0 is reserved for strings (SendString method).
procedure SendBuffer(var Buffer; const BufferSize: Integer; const BufferType: Integer) - Sends data to all the communicators with the same "CommunicationPortID". "Buffer" parameter represents the data you want to send and it can be of any type. "BufferSize" represents the size of the data in bytes you want to send. "BufferType" represents the type of data you want to send and it must be different from 0, because 0 is reserved for strings (SendString method).
function GetAvailableCommunicators(): TStringList - Returns the list of communicator identifiers with the same CommunicationPortID.

Events:
OnReceive: TReceiveEvent = procedure(Sender: TObject; SenderHandle: HWND; BufferType:Integer; Buffer: TStream)- fires when new data was received. The parameter "SenderHandle" represents the identifier of the sender communicator. The parameter BufferType contains the type of received data (0 = AnsiString, user defined data otherwise). The parameter "Buffer" contains the received data.

Usage:
Create two application and put one "ZylAppCommunicator" on each application's main form. Set the "CommunicationPortID" of both communicators to the same value.
You can send data between the communicators using the "SendString" and "SendBuffer" methods and you can use the "OnReceive" method to process the received data.
You can find a demo application in the archived package which shows you the usage of "SendString" and "SendBuffer" methods and OnReceive event. Start the demo in two instances (e.g: one from Delphi and one directly by the included exe file) and you can communicate between the two applications.

Visit product page!

http://www.zylsoft.com
info@zylsoft.com