IPDaemon Control (TM)   Version 1.02

Properties   Events   Error Codes   Copyright & Registration

Description

The IPDaemon Control can be used to create TCP/IP servers running on PC's connected to a TCP/IP network. The control can handle up to 32 simultaneous connections on the same TCP/IP port (service). It is designed to balance the load between connections for a fast, powerful server.

IPDaemon is the server complement of IPPort, which can be used to create client applications. They share a common design philosophy and interface. We expect you will find IPDaemon as easy to use as IPPort.

File Name

IPDAEMON.VBX

Object Type

IPDaemon

Remarks

IPDaemon needs a Winsock 1.1 compliant TCP/IP subsystem. WINSOCK.DLL must be available in the system before the control can be loaded. The Winsock version supported must be at least 1.1.

Each instance of IPDaemon can handle up to 32 simultaneous incoming connections. These connections are identified by a *ConnectionID* (a number between 1 and 32). Most of IPDaemon's properties are array properties. The arrays are indexed by *ConnectionID*. IPDaemon's events also have *ConnectionID* as a parameter to identify the connection they relate to.

Our main goal in designing IPDaemon was ease of use without disregarding performance. The control has a minimum of properties, and five events: ConnectionRequest, Connected, DataIn, Disconnected, ReadyToSend.

IPDaemon can start to listen on a port by setting the Listening property to **True**. When a remote host asks for a connection, the ConnectionRequest is fired. At that point, the connection can either be accepted, or refused. If the connection is accepted, a *ConnectionID* is assigned, and communication can begin. From this point on, the operation is very similar to IPPort. Data is sent by assigning the data string to the DataToSend property. The address and port of the incoming connection can be found by quering the RemoteHost and RemotePort property.

The operation of the control is completely asynchronous. All the calls operate through Windows messages (no blocking calls). The gain in performance is considerable when compared to using blocking calls. The only drawback is what some people perceive as "unnatural" programming, but if you were brave enough to come to this sentence, you will be doing fine.

If you have any questions, suggestions, or need any assistance, you can contact us via email at **devsoft@aol.com**. We will try to answer all messages, however, messages from registered users will have higher priority, so please include your serial number in your message for faster service.

AcceptData~~PREF_AcceptData~~
Active~~PREF_Active~~
BytesSent~~PREF_BytesSent~~
Connected~~PREF_Connected~~
DataIn~~PREF_DataIn~~
DataToSend~~PREF_DataToSend~~
EOL~~PREF_EOL~~
Host~~PREF_Host~~
HostAddress~~PREF_HostAddress~~
HostName~~PREF_HostName~~
InBufferSize~~PREF_InBufferSize~~
Linger~~PREF_Linger~~
Listening~~PREF_Listening~~
LocalHost~~PREF_LocalHost~~
LocalHostName~~PREF_LocalHostName~~
LocalPort~~PREF_LocalPort~~
NullsToSend~~PREF_NullsToSend~~
OutBufferSize~~PREF_OutBufferSize~~
Port~~PREF_Port~~
RemoteHost~~PREF_RemoteHost~~
RemotePort~~PREF_RemotePort~~
WinsockInfo~~PREF_WinsockInfo~~
Action~~PREF_Action~~
EncodedData~~PREF_EncodedData~~
DecodedData~~PREF_DecodedData~~
FileName~~PREF_FileName~~
FileCnt~~PREF_FileCnt~~
FileCnt~~PREF_FileCnt~~
Format~~PREF_Format~~
Intellicode~~PREF_Intellicode~~
MaxFileSize~~PREF_MaxFileSize~~
Overwrite~~PREF_Overwrite~~
ProgressStep~~PREF_ProgressStep~~

Decoding~~UREF_ENCODING~~
Encoding~~UREF_ENCODING~~
UUDecoding~~UREF_UU_ENCODING~~
UUEncoding~~UREF_UU_ENCODING~~
Base64 Decoding~~UREF_BASE64_ENCODING~~
Base64 Encoding~~UREF_BASE64_ENCODING~~
Quoted Printable Decoding~~UREF_QP_ENCODING~~
Quoted Printable Encoding~~UREF_QP_ENCODING~~

Connected~~EREF_Connected~~
ConnectionRequest~~EREF_ConnectionRequest~~
DataIn~~EREF_DataIn~~
Disconnected~~EREF_Disconnected~~
ReadyToSend~~EREF_ReadyToSend~~
Progress~~EREF_Progress~~

Encode~~FREF_Encode~~
Decode~~FREF_Decode~~

True

False
Boolean (Integer)
Integer
Long
String
"" *(empty string)*

".uue", **".b16"**, or **".q_p"**

Error Codes~~ERROR_CODES~~
Exported Functions~~EXPORTED_FUNCTIONS~~

IPDaemon

IPDAEMON.VBX

ipdaemoncontrol

1.0

$25

#11776

4441

Copyright Notice

Registration Procedure

The prices below are for the ***licenses only*** and do not include media distribution. We only send you a set of keys to unlock the software and verify registration by e-mail. All technical support questions should be directed to:

INTERNET:	**devsoft@aol.com**
COMPUSERVE:	**75244,2736**

The cost of a single user developer is **$25**. You can order via any one of the following channels:

i) ordering through CompuServe Software Registration Service (SWREG)

You can register via CompuServe by going to the Shareware Registration Forum (**GO SWREG**) and following the forum instructions. The Registration ID for **IPDaemon** is **4441**. You can also do a keyword search using the keyword **IPDaemon**.

ii) ordering by Check or Money Order

To order by check or money order please send the attached order form and a check or a money order **(payments must be in US Dollars drawn on a US Bank)** to:

devSoft Inc.
P.O. Box 13821
Research Triangle Park, NC 27709 U.S.A.

iii) ordering by Credit Card

To order by Visa or MasterCard by E-mail, fax or snail mail send the attached order form to the above address or:

INTERNET:	**devsoft@aol.com**
COMPUSERVE:	**75244,2736**
FAX:	**(919) 493-5805**

Where To Find Our Shareware

The first place to look at is **http://www.dev-soft.com/devsoft .** There you will find the latest versions of our products, release notes, questions and answers, documentation, press releases, everything you would want to know about us and our products. We strongly recommend that you access that site before contacting us directly.

We will also upload our products in the CompuServe MSBASIC Forum (GO MSBASIC) in Library 17 (3rd Party Tools), as well as in America Online. Usually, the name of the product will be listed as a keyword, so if you try it, you will certainly get a hit.

We will also announce our new releases to the newsgroups of the comp.lang.basic.visual hierarchy, and **comp.lang.basic.visual.3rdparty** in particular.

Licensing

i) shareware version

You may use the shareware version of **IPDaemon** for up to 30 days in your design environment and for evaluation purposes only. You may copy and distribute it freely as long as all the files in the package, including the demo programs and this help file are distributed with it and no changes or additions of any kind are made to the original package.

There is no charge for any of the above, however, you are specifically prohibited from charging, or requesting donations for any copies, however made, and from distributing **IPDaemon** and/or it's

devSoft Inc.

P.O. Box 13821 , Research Triangle Park, NC 27709 U.S.A.

ORDER FORM / INVOICE

Prices are guaranteed through December 1995.

Registration codes will be sent by electronic mail only. If you need a disk (3.5") or a paper copy of your license, please enclose an additional $5.00 with your order.

Product	**No. of Copies**	**Price**	**Total**
IPPort	__________ x	$25.00 =	__________
IPDaemon	__________ x	$25.00 =	__________
UDPPort	__________ x	$25.00 =	__________
NetCode	__________ x	$30.00 =	__________
	Add Disk and/or Paper copy of License	$5.00 =	__________
		Total	__________

Name: Phone: Date:

Credit card used /Exp. Date

Company Name:

Address:

City, State, Zip/Country:

Name of registrant:

E-Mail address:

Comments:

Properties

*AcceptData	*Listening
*BytesSent	*LocalHostName
*Connected	Name
*DataToSend	*OutBufferSize
*EOL	*Port
Index	*RemoteHost
*InBufferSize	*RemotePort
Left	Top
*Linger	*WinsockInfo

AcceptData Property

Description

Setting the property to **False** temporarily disables data reception (and the DataIn event) from the specified *ConnectionID*. Setting the property to **True** reenables data reception.

Usage

[*form.*][*ipdaemoncontrol.*]**AcceptData(***ConnectionID***)**[= *value*]

Default Value

True

Remarks

Use the **AcceptData** property with caution. If data reception is disabled for too long, the other side might abort the connection.

This property is not available in design mode.

Data Type

Boolean (Integer)

BytesSent Property

Description

Shows the number of bytes sent after the last assignment to the DataToSend property.

Usage

[*form*.][*ipdaemoncontrol*.]**BytesSent**

Default Value

0

Remarks

BytesSent shows how many bytes were sent after the last assignement to DataToSend. Check the DataToSend property for more information.

Since BytesSent is shared among live connections, it's value must be read immediately after the assignement to DataToSend, or it's value might change because of a send in another *ConnectionID*.

This property is read-only and not available in design mode.

Data Type

Integer

Connected Property

Description

This property shows whether a *ConnectionID* is valid (connected) or not. Setting it to **False** closes the connection corresponding to *ConnectionID*.

Usage

[*form*.][*ipdaemoncontrol*.]**Connected(***ConnectionID***)**[= *value*]

Default Value

False

Remarks

Connected is an action property. Use it to close connections.

Setting **Connected** to **True** will generate an error (connections are initiated only by a remote host).

How and when the connection is closed is controlled by the Linger property. Please refer to its description for more information.

The **Connected** property is not available in design mode.

Data Type

Boolean (Integer)

DataToSend Property

Description

DataToSend is an action property. Assigning a Visual Basic string to this property makes the control send the string to the remote host (note that a Visual Basic string can contain control as well as NULL characters).

Usage

[*form*.][*ipdaemoncontrol*.]**DataToSend**(*ConnectionID*)[= *value*]

Default Value

"" *(empty string)*

Remarks

If you are sending data to the remote host faster than it can process it, or faster than the network bandwidth allows, the outgoing queue might fill up. When this happens, **DataToSend** fails with error 25036: "[10035] Operation would block" (WSAEWOULDBLOCK). The BytesSent property shows how many bytes were sent (if any). You can trap the error, and then try to send the data again. If 0 bytes were sent, then you can wait for the ReadyToSend event before attempting to send data again. (However, please note that ReadyToSend is not fired when part of the data are successfully sent).

This property is write-only and not available in design mode.

Data Type

String

EOL Property

Description

Used to break the incoming data stream into chunks separated by the string assigned to **EOL**. For more information, see the description of the DataIn event.

Usage

[*form*.][*ipdaemoncontrol*.]**EOL**[= *value*]

Default Value

"" *(empty string)*

Remarks

The **EOL** property is especially useful with ASCII files. Setting it to **Chr$(10)** (*newline*) enables splitting of an incoming ASCII text stream into lines. In this case, one event is fired for each line received (as well as in packet boundaries). The **Chr$(10)** characters are discarded.

EOL is a Visual Basic String. In particular, this means that it can be more than one character long, and it can contain NULL (0) characters as well.

The **EOL** property is shared among incoming connections.

Data Type

String

InBufferSize Property

Description

Specifies the size (in bytes) of the receiving queue in the underlying TCP/IP provider.

Usage

[*form*.][*ipdaemoncontrol*.]**InBufferSize**[= *value*]

Default Value

2048

Remarks

This is the size of an internal queue in the TCP/IP provider. You can increase or decrease its size depending on the amount of data that you will be receiving. Increasing **InBufferSize** can provide drastic improvements in performance in some cases.

This property is shared among connections. It takes effect when a new connection is accepted.

Some TCP/IP implementations do not support variable buffer sizes. If that is the case, when a new connection is accepted, **InBufferSize** reverts back to its allowable size. The same happens if you attempt to make it too large or too small.

Data Type

Integer

Linger Property

Description

This property controls how a connection is closed. The default is **True**. In this case the connection is closed only after all the data is sent. Setting it to **False** forces an abrupt (hard) disconnection. Any data that were in the sending queue might be lost.

Usage

[*form*.][*ipdaemoncontrol*.]**Linger**[= *value*]

Default Value

True

Remarks

The **Linger** property is shared among connections. It's value controls how the next connection will be closed (if IPDaemon is closing the connection and not the remote host).

The default behaviour (which is also the default mode for Winsock stream sockets) might result in an indefinite delay in closing the connection. Even though IPDaemon returns control immediately, Winsock might indefinitely hold system resources until all pending data are sent (even after your application closes). This means that valuable system resources might be wasted.

Setting **Linger** to **False** forces an immediate disconnection. If you know that the other side has received all the data you had sent (by a client acknowledment, for example), setting **Linger** to **False** might be the appropriate course of action.

Data Type

Boolean (Integer)

LocalHostName Property

Description

Specifies the domain name of the local host.

Usage

[*form*.][*ipdaemoncontrol*.]**LocalHostName**

Default Value

"" *(empty string)*

Remarks

This property is read-only.

Data Type

String

Listening Property

Description

This property is used to enable IPDaemon to accept connections on a port.

Usage

[*form*.][*ipdaemoncontrol*.]**Listening**[= *value*]

Default Value

False

Remarks

Listening is an action property. Use it to make IPDaemon *'listen'* to the port specified by the Port property. Setting **Listening** to **False** will make IPDaemon stop listening. (Please note that this does not close the existing connections).

The **Listening** property is not available in design mode.

Data Type

Boolean (Integer)

OutBufferSize Property

Description

Specifies the size (in bytes) of the outgoing queue in the underlying TCP/IP provider.

Usage

[*form*.][*ipdaemoncontrol*.]**OutBufferSize**[= *value*]

Default Value

2048

Remarks

This is the size of an internal queue in the TCP/IP provider. You can increase or decrease its size depending on the amount of data that you will be receiving. Increasing **OutBufferSize** can provide drastic improvements in performance in some cases.

This property is shared among connections. It takes effect when a new connection is accepted.

Some TCP/IP implementations do not support variable buffer sizes. If that is the case, when a new connection is accepted, **OutBufferSize** reverts back to its allowable size. The same happens if you attempt to make it too large or too small.

Data Type

Integer

Port Property

Description

This is the port number IPDaemon listens to for incoming connections.

Usage

[*form*.][*ipdaemoncontrol*.]**Port**[= *value*]

Default Value

0

Remarks

The **Port** property must be set before IPDaemon starts listening. If it's value is 0, then the TCP/IP subsystem picks a port number at random. It's number can be found by checking the value of the **Port** property after IPDaemon is in listening mode (after succesfully assigning **True** to the Listening property).

The service port is not shared among servers (i.e. there can be only one IPDaemon *'listening'* on a particular port).

Data Type

Integer

RemoteHost Property

Description

Specifies the remote host IP number in Internet dotted format.

Usage

[*form*.][*ipdaemoncontrol*.]**RemoteHost(***ConnectionID***)**[= *value*]

Default Value

0.0.0.0

Remarks

ConnectionID must indicate a valid connection, or an error will be fired.

This property is read-only and not available in design mode.

Data Type

String

RemotePort Property

Description

Specifies the IP port of the remote host.

Usage

[*form*.][*ipdaemoncontrol*.]**RemotePort(***ConnectionID***)**[= *value*]

Default Value

0

Remarks

ConnectionID must indicate a valid connection, or an error will be fired.

This property is read-only and not available in design mode.

Data Type

Integer

WinsockInfo Property

Description

Provides information about the underlying TCP/IP (Winsock) provider.

Usage

[*form*.][*ipdaemoncontrol*.]**WinsockInfo**

Default Value

"" *(empty string)*

Remarks

WinsockInfo returns a string up to 256 bytes long provided by the underlying Winsock subsystem.

If Winsock fails to initialize successfully, **WinsockInfo** contains the string "Not Initialized." followed by a description of the error condition.

The property is read-only.

Data Type

String

Events

*ConnectionRequest

*Connected

*DataIn

*Disconnected

*ReadyToSend

Connected Event

Description

Occurrs after a connection is accepted from a remote host.

Syntax

Sub *ipdaemoncontrol*_**Connected(***ConnectionID* **As Integer**, *StatusCode* **As Integer**, *Description* **As String)**

Remarks

If a connection is successfully created, *StatusCode* is 0, and *Description* is "**OK**".

If the connection fails, *StatusCode* has the error code returned by Winsock. *Description* contains a description of this code. The value of *StatusCode* is obtained by adding 15001 to the corresponding Winsock error code.

Please refer to the Error Codes section for more information.

ConnectionRequest Event

Description

Occurs when a remote host attempts to connect to IPDaemon.

Syntax

Sub *ipdaemoncontrol*_**ConnectionRequest(***Accept* **As Integer)**

Remarks

This event indicates an incoming connection. The connection is accepted by default. If you want to refuse it, you can set the *Accept* parameter to **False**.

DataIn Event

Description

Occurs when data arrives from the remote host.

Syntax

Sub *ipdaemoncontrol*_**ConnectionRequest(***ConnectionID* **As Integer,** *Text* **As String,** *EOL* **As Integer)**

Remarks

Trapping the **DataIn** event is your only chance to get the data coming from the other end of the connection. The incoming data are given in *Text*. *Text* is a Visual Basic string, and as such might be considered as a binary chunk of data with length **Len(***Text***)**.

EOL indicates whether the EOL string was found on the end of *Text* or not. If the EOL string was found, than *EOL* is **True**.

If *Text* was obtained at the end of a segment of data received from Winsock, then *EOL* is **False**. Please note that this also means that more than one **DataIn** event with *EOL* set to **False** can be received during a connection.

If the EOL property is "" (empty), then *EOL* can be disregarded. For more information on *EOL,* please refer to the description of the EOL property.

Disconnected Event

Description

Occurs when the connection to the remote host is closed (broken).

Syntax

Sub *ipdaemoncontrol*_**Disconnected(***ConnectionID* **As Integer**, *StatusCode* **As Integer**, *Description* **As String)**

Remarks

If the connection is broken normally, *StatusCode* is 0, and *Description* is "**OK**".

If the connection is broken for any other reason, *StatusCode* has the error code returned by Winsock. *Description* contains a description of this code. The value of *StatusCode* is obtained by adding 15001 to the corresponding Winsock error code.

Please refer to the Error Codes section for more information.

ReadyToSend Event

Description

Indicates that the underlying TCP/IP subsystem is ready to accept data and send them to the remote host.

Syntax

Sub *ipdaemoncontrol*_**ReadyToSend(***ConnectionID* **As Integer)**

Remarks

The **ReadyToSend** event is fired when the connection is ready to accept data again after a failed DataToSend.

The event is also fired immediately after a connection is accepted.

Error Codes

The following is a list of the trappable errors fired by IPDaemon:

IPDaemon Internal Errors

20106	Winsock error code outside normal range.
20107	You cannot change the Port while **IPDaemon** is listening.
20127	Invalid *ConnectionID*.

Winsock Errors

The error message descriptions show the corresponding Winsock error number. The corresponding Visual Basic error code can be obtained by adding 15001 to the number displayed in the message and vice-versa.

25005	(WSAEINTR)	[10004] Interrupted system call.
25010	(WSAEBADF)	[10009] Bad file number.
25014	(WSAEACCES)	[10013] Permission denied.
25015	(WSAEFAULT)	[10014] Bad address.
25023	(WSAEINVAL)	[10022] Invalid argument.
25025	(WSAEMFILE)	[10024] Too many open files.
25036	(WSAEWOULDBLOCK)	[10035] Operation would block.
25037	(WSAEINPROGRESS)	[10036] Operation now in progress.
25038	(WSAEALREADY)	[10037] Operation already in progress.
25039	(WSAENOTSOCK)	[10038] Socket operation on non-socket.
25040	(WSAEDESTADDRREQ)	[10039] Destination address required.
25041	(WSAEMSGSIZE)	[10040] Message too long.
25042	(WSAEPROTOTYPE)	[10041] Protocol wrong type for socket.
25043	(WSAENOPROTOOPT)	[10042] Bad protocol option.
25044	(WSAEPROTONOSUPPORT)	[10043] Protocol not supported.
25045	(WSAESOCKTNOSUPPORT)	[10044] Socket type not supported.
25046	(WSAEOPNOTSUPP)	[10045] Operation not supported on socket.
25047	(WSAEPFNOSUPPORT)	[10046] Protocol family not supported.
25048	(WSAEAFNOSUPPORT)	[10047] Address family not supported by protocol family.
25049	(WSAEADDRINUSE)	[10048] Address already in use.
25050	(WSAEADDRNOTAVAIL)	[10049] Can't assign requested address.
25051	(WSAENETDOWN)	[10050] Network is down.
25052	(WSAENETUNREACH)	[10051] Network is unreachable.
25053	(WSAENETRESET)	[10052] Net dropped connection or reset.
25054	(WSAECONNABORTED)	[10053] Software caused connection abort.
25055	(WSAECONNRESET)	[10054] Connection reset by peer.
25056	(WSAENOBUFS)	[10055] No buffer space available.
25057	(WSAEISCONN)	[10056] Socket is already connected.
25058	(WSAENOTCONN)	[10057] Socket is not connected.
25059	(WSAESHUTDOWN)	[10058] Can't send after socket shutdown.
25060	(WSAETOOMANYREFS)	[10059] Too many references, can't splice.
25061	(WSAETIMEDOUT)	[10060] Connection timed out.
25062	(WSAECONNREFUSED)	[10061] Connection refused.
25063	(WSAELOOP)	[10062] Too many levels of symbolic links.
25064	(WSAENAMETOOLONG)	[10063] File name too long.
25065	(WSAEHOSTDOWN)	[10064] Host is down.

25066	(WSAEHOSTUNREACH)	[10065] No Route to Host.
25067	(WSAENOTEMPTY)	[10066] Directory not empty.
25068	(WSAEPROCLIM)	[10067] Too many processes.
25069	(WSAEUSERS)	[10068] Too many users.
25070	(WSAEDQUOT)	[10069] Disc Quota Exceeded.
25071	(WSAESTALE)	[10070] Stale NFS file handle.
25072	(WSAEREMOTE)	[10071] Too many levels of remote in path.
25092	(WSASYSNOTREADY)	[10091] Network SubSystem is unavailable.
25093	(WSAVERNOTSUPPORTED)	[10092] WINSOCK DLL Version out of range.
25094	(WSANOTINITIALISED)	[10093] Successful WSASTARTUP not yet performed.
25102	(WSAHOST_NOT_FOUND)	[11001] Host not found.
25103	(WSATRY_AGAIN)	[11002] Non-Authoritative Host not found (try again).
25104	(WSANO_RECOVERY)	[11003] Non-Recoverable error.
25105	(WSANO_DATA)	[11004] Valid name, no data record for requested name.