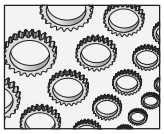
CHAPTER

20

Overview of Internet Direct

Internet Direct consists of a set of about 70 components for Internet development in Borland's Kylix and Delphi products. These components include client components, server components, and various supporting components for writing these clients and servers. Internet Direct differs a bit from the Kylix product itself in that all of the components included with Internet Direct are open source. You will read more about this later in this chapter.

This chapter provides an overview of Internet Direct and describes its major features and characteristics. You will also find a description of all of the Internet Direct components that are included with Kylix. Next, discussions of how clients work and how servers work are included. Finally, licensing requirements for using Internet Direct components in your Kylix applications and getting technical support are discussed.



NOTE

If you are not familiar with Internet-related technologies and terms, you may want to read through Chapter 17 first, or refer back to Chapter 17 periodically as you read about Internet Direct.

What Is Internet Direct?

Internet Direct, or Indy for short, is a set of open source socket components that are included with Kylix, as well as with Delphi 6. Internet Direct consists of clients, servers, and support components. These Internet components include both client and server implementations of popular Internet protocols such as SMTP (Simple Mail Transfer Protocol), Telnet, and FTP (File Transfer Protocol).

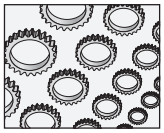
One of the more important features of Internet Direct is that it is compatible with a wide variety of development tools that you might be using, including Kylix, Delphi 4, Delphi 5, Delphi 6, C++ Builder 4, and C++ Builder 5. This means that you can easily take application code that makes use of Internet Direct components and reuse it with other applications created in Delphi and C++ Builder.

Internet Direct is both powerful and easy to use. The following is a list of some of its more important features:

- The Internet Direct source code is cross-platform and single source—all operating system differences are handled internally and transparently at a low level by Internet Direct. Currently, both Linux and Windows are supported, and the source code includes very few conditional compiler statements ({$IFDEF}). In most cases, code that you write for Kylix will run unchanged under Delphi (so long as you are using Internet Direct in Delphi). Keep in mind, however, that it is always advisable to thoroughly test your code under all operating systems you plan your code to run under.

- Many of the low-level implementation details are encapsulated in the Internet Direct components, including such things as the TCP/IP stack. This permits you to concentrate on your application's needs without having to fiddle with the minutiae of socket implementation.
- Internet Direct provides support for multibyte characters sets (MBCS) including Asian character sets. For example, Internet Direct supports Japanese, Chinese, Korean, and other sets.
- Internet Direct includes both base-level socket communication protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Internet Direct also supports a large selection of specific protocol implementations such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP). While Internet Direct does not require you to understand the low-level details of the various protocol implementations, it is generally a good idea for you to have a basic familiarity with the particular protocol you are using. If you are unfamiliar with a particular protocol, you can look up the official RFC (Request for Comments) documents of the IETF (Internet Engineering Task Force) that describe protocols in detail. You can find most of the relevant RFCs by visiting http://www.ietf.org or http://www.w3c.org.
- Internet Direct is open source, providing both a modified Berkeley Software Distribution (BSD) public license and a Mozilla Public License (MPL). Internet Direct includes full source code for components, and, since it's open source, you can use and modify this source for your particular needs. However, because Internet Direct is open source, you must follow Internet Direct's licensing policies and include copyright and licensing information in your source code that utilizes Internet Direct components. Internet Direct licensing requirements are described in greater detail at the end of this chapter.
- Internet Direct is thread based, which is invaluable in the concurrent world or client/server applications. For example, because Internet Direct server components support threading, server applications built with Internet Direct can easily handle many simultaneous client requests.
- Internet Direct supports thread pooling. Under normal circumstances, the creation and destruction of threads is a resource-intensive process. This is especially true with servers that have short-lived connections. Such servers create a thread, use it for very brief time, and then destroy it. Without thread pooling, such servers spend a relative large amount of time creating and destroying threads. Thread pooling in Internet Direct can be achieved via use of the TIdThreadMgrPool component.
- Internet Direct provides both SOCKS4 and SOCKS5 proxy support. Socks is a very common proxy protocol, but it is not a transparent proxy. To use a SOCKS-type proxy, applications must support the SOCKS proxy protocol.

- Internet Direct is based entirely on blocking socket calls. A blocking call is one that behaves similar to a Pascal function invocation, in that the call pauses, or blocks, until the call is complete. Writing clients and servers that use blocking calls is much easier than writing those that use nonblocking calls.
- Because Internet Direct does not rely on any messaging system, Windows messaging, or any part of the X Windows system, Internet Direct can be used to create console-based clients and servers. Console applications are extremely important in Linux, as the X Windows system is optional and not installed on most Linux servers. Internet Direct also has many status methods and other mechanisms that make it easy to create Internet Direct–based X Windows applications as well. The demos that come with Internet Direct demonstrate both types of applications.
- Updates to Internet Direct components can be downloaded from the Web at the Indy Web site, which you can find at http://www.nevrona.com/Indy/. Also available for download from this Web site are complete versions of documentation and demos for Internet Direct.



NOTE

Internet Direct is an outgrowth of the third-party component set named Winshoes. Winshoes was originally developed by Chad Z. Hower (also known as "Kudzu"). Internet Direct is Winshoes version 8 and was developed by the Indy Pit Crew under the leadership of Chad Z. Hower. Both products are distributed as open source and sponsored by Nevrona Designs. As it was being designed, one of the major goals of Winshoes 8 was Kylix compatibility. Since Kylix is a development tool for Linux, the name "Win"shoes was no longer appropriate. (The name "Winshoes" was a play on the name of the Windows sockets, named winsock.)

Internet Direct Components

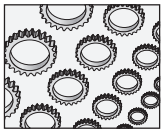
Indy components in Kylix are grouped into three categories:

- Client components
- Server components
- Support components

Each group of components appears on a separate page of the Component Palette. These pages are named:

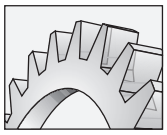
- Indy Clients
- Indy Servers
- Indy Misc

The following sections contain a brief summary of each category of the Internet Direct components that appear on the Component Palette. However, it is worth noting that Internet Direct actually consists of many more component and classes than those that are visible on the Component Palette. Specifically, a few of the more interesting Internet Direct classes are internal or serve as helper classes. TIdThread and TIdEMailAddressItem are examples of such components. Chapter 21 includes some examples that demonstrate the use of some of these other classes.



NOTE

You can find additional information about individual Internet Direct components using the online help. The online help provides descriptions and some code examples along with the syntax for each of the components, so the syntax for the components is not duplicated here. The online help also includes a component's properties, its methods, and the unit in which the class is declared. To access online help for Internet Direct, simply move your cursor over the desired component and press F1.



CAUTION

In the initial release of Kylix (version 1.0), some of the links in the online help for information about Internet Direct do not work correctly. Also, the `.tgz` *file that contains the Indy Demos that ships with Kylix 1.0 has some distribution problems. Some files were omitted in Borland's* `.tgz`*, and other files had their name's case changed, which has caused nearly all the demos not to compile. A corrected* `.tgz` *of the Indy Demos is available for download at http://www.nevrona.com/indy/download80.html.*

Indy Client Components

The Indy Clients page shown here contains general-purpose TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) client components, as well as client components that implement clients for specific protocols (FTP, Echo, and so forth).

III 20-1

Components on the Indy Clients page are used to write socket clients. A socket client requests a connection with a server. If the connection is accepted, the client

and server can begin communicating using the connection. This communication can be of any nature but is typically defined by a protocol (either an established protocol such as HTTP or one you make up). A protocol (as described in Chapter 17) is nothing more than a set of rules that two sockets use to communicate.

Most of the clients have corresponding server components (you can see this by the similarity in their names), and vice versa, but not all do. Most of the clients implement a specific protocol, but some do not. For example, the TIdEcho client implements the Echo protocol. On the other hand, the TIdTCPClient component only supports TCP, which is used by many more specific protocols.

The following descriptions of the individual client components also include descriptions of the corresponding protocols for the client component in question. The protocols are described here, although each protocol involves at a minimum both a client and a corresponding server component.

TIdTCPClient

TIdTCPClient is the base TCP (Transmission Control Protocol) client component. This component is used for basic TCP communication. It is also the base component for all implemented protocols that use TCP such as TIdSMTP and TIdFTP.

TIdUDPClient

TIdUDPClient is the base UDP (User Datagram Protocol) client component. This component is used for basic UDP communication. It is also the base component for all implemented protocols that use UDP.

TIdDayTime

DayTime client is inherited from TIdTCPClient. It is used to connect to a Day Time server to retrieve the current date and time.

TIdDNSResolver

TIdDNSResolver is a DNS (Domain Name Service) client that allows DNS queries to be built and requested from a DNS server. Queries to the DNS server consist of a query package that indicates the domain and the records that are required. It is based on UDP.

TIdEcho

TIdEcho is a simple Echo client used to connect to an Echo server. Echo servers respond with the same text that is sent to them, thus the name echo.

TIdFinger

TIdFinger is a Finger client used to retrieve information from a Finger server. Finger is a protocol that allows the user to request data about a particular user on a system.

TIdFTP

TIdFTP is a full-blown FTP (File Transfer Protocol) client that supports both passive and active transfers. All normal FTP operations are included, such as GET and PUT, as well as other specific ones such as deleting directories and getting quotas and sizes of files and directories.

TIdGopher

TIdGopher is a Gopher client component. Gopher is a protocol that has been for the most part supplanted by the World Wide Web (HTTP).

TIdHTTP

TIdHTTP is an HTTP Client supporting both version 1.0 and 1.1. It includes GET, POST, and HEAD operations. Proxy and authentication are also supported.

TIdIcmpClient

TIdIcmpClient is a base ICMP (Internet Control Message Protocol) client component. This component is used for basic ICMP communication. It is used to implement protocols that use ICMP, such as ping and traceroute.

TIdPOP3

TIdPOP3 is a POP3 client that supports the typical POP (Post Office Protocol) operations. TIdPOP3 supports MIME decoding and multibyte character sets.

TIdNNTP

TIdNNTP is an NNTP (Network News Transfer Protocol) client that includes support for MIME encoding and decoding, multibyte character sets, and alternative types.

TIdQOTD

TIdQOTD is a Quote of the Day client and is used to connect to the corresponding server to retrieve daily quotes. These are not stock quotes, but quotes such as "Programming is an art form that fights back." Each quote server contains its own unique database of quotes.

TIdSMTP

TIdSMTP is an SMTP (Simple Mail Transfer Protocol) client that includes support for authentication, MIME encoding, and multibyte character sets.

TIdSNTP

TIdSNTP is an SNTP (Simple Network Time Protocol) client. TIdSNTP can be used to connect to any time server to retrieve the current time of day.

TIdTelnet

TIdTelnet is a Telnet client that used for creating remote sessions on another computer. TIdTelnet includes console negotiation and authentication. (Telnet is a protocol that usually has a live person interacting as the client.) TIdTelnet does not provide for the display or terminal emulation. It merely implements the communication portions for communicating with a Telnet server.

TIdTime

The TIdTime client provides an alternative to using the TIdSNTP client to retrieve the time. It is important to note that the format of the returned data is different from that of SNTP. TIdTime is based on RFC 868 and retrieves the time format internally in Unix format. TIdTime performs all the necessary conversions from Unix time.

TIdTrivialFTP

The TIdTrivialFTP client is based on the Trivial File Transfer Protocol. This client can be used to connect to TFTP servers. TFTP is not for general file transfer, since it is a very lightweight file transfer protocol. It is usually limited to LANs and used for simple tasks such as uploading/downloading routing tables from routers. Due to the nature of this protocol, it is not recommended to use when authentication is required or any type of security in general is needed.

TIdWhois

TIdWhois is a Whois client that you use to connect to a Whois server. By connecting to any standard Whois server, information about domains can be retrieved.

Indy Server Components

The Indy Servers page, shown here, contains general-purpose TCP and UDP servers, as well as basic interfaces for building specific protocol servers. Servers, as

described in Chapter 17, are applications that listen for a connection request from another application, referred to as the client.

Ill 20-2

The following sections provide a brief description of each of the Internet Direct server components.

TIdTCPServer

The TIdTCPServer component is used for building basic TCP servers. It is also the base component for all implemented protocol server components that use TCP, such as TIdHTTPServer and TIdNNTPServer.

TIdUDPServer

TIdUDPServer is used for all UDP-based servers, such as the Trivial FTP Server.

TIdChargenServer

TIdChargenServer is a CharGen server that is used to generate random characters. This component can be used to implement such a service. TIdChargenServer is usually only used for testing purposes.

TIdDayTimeServer

TIdDayTimeServer is the counterpart of the Day Time client component, TIdDayTime. It is used to develop a Day Time server that will respond to client connections with the current date and time.

TIdDICTServer

TIdDictServer implements the interface for implementing a dictionary server. The Dictionary Server Protocol (DICT) is a TCP transaction-based query/response protocol that allows a client to access dictionary definitions from a set of natural language dictionary databases.

TIdDISCARDServer

TIdDiscardServer implements a discard server. Discard is a useful debugging and measurement tool. A discard service simply throws away any data it receives.

TIdECHOServer

TIdECHOServer is an Echo server component. Echo servers respond with the same text that is sent to them, thus the name echo. They are primarily used for testing purposes.

TIdFingerServer

As mentioned previously, some servers allow requests from clients about users that are present on the system. Known as a Finger server, this TIdFingerServer component can be used to implement such a service.

TIdGopherServer

A Gopher server is a distributed document system for presenting documents through a hierarchical system of menus. Although mostly replaced by the World Wide Web, Gopher servers are still used for accessing some legacy systems and have lower overhead than a Web browser. This TIdGopherServer component can be used to implement such a service.

TIdHostNameServer

TIdHostNameServer is a Hostname server component. A Hostname server is used to respond with information about particular hostnames.

TIdHTTPServer

TIdHTTPServer is used for implementing an HTTP (Web) server. It is important to understand that the component by itself is not a Web server. It is used to develop a Web server or any other server that runs using HTTP as the protocol. However, many features are included with this component that make this task much easier, such as authentication, cookies, and state management.

TIdIMAP4Server

The TIdIMAP4Server component implements the basic operations that can be performed on an IMAP (Internet Message Access Protocol) server. It allows the retrieval of e-mail messages from a server. The primary difference between IMAP and POP is that, whereas POP can be thought of as temporary storage, IMAP is focused on mail and folders being stored and accessed on the server instead of the client machine. IMAP4 was intended as the replacement for the POP3 protocol. However, it is not widely used and POP3 is still the common standard.

TIdIRCServer

One of the most common services on the Internet today, and commonly known as "chat," the TIdIRCServer component provides the basic building blocks for an IRC (Internet Relay Chat) server.

TIdMappedPortTCP

The TIdMappedPortTCP component allows for easy construction of mapped ports, which are often used in proxies. The basics behind this component are to map one port to another. For example, port 80 could be mapped to port 3000. Then, all requests to this first port (port 80) would be redirected to the latter port (port 3000).

TIdNNTPServer

TIdNNTPServer is an NNTP (Network News Transfer Protocol) server component that allows the creation of a news server. It is important to note that this is not a full-blown news server but a component that provides the basic functionality required by a news server.

TIdQOTDServer

The TIdQOTDServer is the counterpart to the client component, TIdQOTDClient. The Quote of the Day server allows the implementation of a service that provides quotes.

TIdSimpleServer

TIdSimpleServer is a lightweight TCP server that allows acceptance of a single inbound connection. This is used for establishing individual data channels in the FTP protocol and other areas.

TIdTelnetServer

TIdTelnetServer implements the interface for implementing a Telnet server. Telnet is a protocol that usually has a live person interacting as the client. A Telnet server is useful for creating remote text-based interfaces for people, as opposed to a protocol that uses an application for the client.

TIdTimeServer

Similar in functioning to the Day Time server, this simple component can be used to offer a time service on a machine. No additional code is required. Just activating the TIdTimeServer component is sufficient to return the time based on the internal clock of the server machine.

TIdTrivialFTPServer

TIdTrivialFTPServer provides a Trivial File Transfer Protocol service. As mentioned previously in the section describing the corresponding TIdTrivialFTP client component, this service is hardly ever used due to the lack of error checking and security. Mainly it is used to transfer files to a hardware device for the purpose of chipset updates.

TIdTunnelMaster and TIdTunnelSlave

The tunnel components, TIdTunnelMaster and TIdTunnelSlave, are used to proxy multiple connections across a single connection (the tunnel). These components can be used for a variety of purposes such as securing communications over an insecure connection.

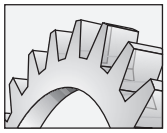
TIdWhoIsServer

TIdWhoIsServer implements the basic functionality provided by NIC servers, responding with information regarding domains requested by a Whois client.

Indy Miscellaneous Components

Components that appear on the Indy Misc page of the Component Palette shown here include support components for client and server components as well as extra components such as coders (encoders and decoders), encryption components, thread managers, and information components. These components are described in the following sections.

III 20-3



CAUTION

The encoders and decoders that shipped with the original version of Kylix were designed for internal use only, and often proved difficult to use stand-alone. Updated versions of the coders are available in Internet Direct version 8.1, which can be downloaded from the Indy Web site.

TIdAntiFreeze

Due to the blocking nature of Internet Direct components, the user interface normally gives the impression that it has frozen while a communication is in

progress. Instead of using secondary threads for the communication, you can place an IdAntiFreeze component onto your form to prevent the user interface from freezing. More information about TIdAntiFreeze and how it operates is offered later, in the section "Using TIdAntiFreeze."

TIdDateTimeStamp

TIdDateTimeStamp is a class for performing mathematics, conversions, and time zone operations on dates and times. The Internet protocols use many different date and time formats. In addition, the Internet is a worldwide network and covers all time zones. TIdDateTimeStamp encapsulates many of the operations necessary for handling these needs.

TIdIPWatch

IP Watch is a timer-based component that constantly monitors changes in the IP (Internet Protocol) address of the computer. Events can be assigned to trigger specific actions when an IP change is detected. This component is commonly used to detect when a computer is connected to the Internet (or connected to any network, in general). The change in IP address, in this situation, would be due to the assignment of an IP address from a DHCP (Dynamic Host Configuration Protocol) server upon connection to the new network. Typically, this is caused by the dialing up of a modem or other nondedicated connection.

TIdLogDebug

By assigning a TIdLogDebug component to an Intercept property of any client or server component, the communication process can be logged to a specified file. This component is very useful for debugging Internet Direct components.

TIdMessage

TIdMessage is used in combination with other components to properly decode or encode messages. This can be a POP, SMTP, or NNTP component. It supports MIME encoding and decoding, multibyte character sets, and ISO (International Organization for Standardization, translated) encoding.

TIdNetworkCalculator

One of the few Internet Direct components that is entirely operable at design time, the Network Calculator can be used to calculate anything to do with IP addresses, including netmasks, subnet, network classes, and so on.

TIdThreadMgrDefault

TIdThreadMgrDefault is the default thread manager. If no TIdThreadManager is specified for a component that supports thread management, an implicit instance of TIdThreadManagerDefault is created. TIdThreadManagerDefault provides only basic thread management and creates and destroys threads on demand.

TIdThreadMgrPool

TIdThreadMgrPool is a more advanced thread manager than TIdThreadMgrDefault. TIdThreadMgrPool is more efficient because it does not create and destroy threads on demand but pools them instead.

TIdVCard

A VCard is the electronic equivalent of a business card. It can include contact information, logos, graphics, and personal information about the creator (owner). The TIdVCard component can be used to load, save, parse, and manipulate VCards.

TidQuotedPrintableEncoder (Decoder)

Used by IdMessage, the QuotedPrintableEncoder allows the encoding (decoding) of text in quoted-printable format. It can be used as a stand-alone component also if this type of encoding (decoding) is required. Quoted-printable is an algorithm that permits the transportation of messages containing nonprintable ASCII characters.

TIdBase64Encoder (Decoder)

Primarily used by IdMessage to encode (decode) attachments, this is another form of encoding algorithm that allows transparent transportation of messages with nonprintable characters.

TIdUUEncoder (Decoder)

One of the first encoding algorithms, UU encoding (UUCP refers to Unix-to-Unix Copy Program) has never become a standard and is only rarely used when sending attachments with articles posted to news servers.

TIdXXEncoder (Decoder)

This method of encoding is hardly ever used. It is the same as UU encoding but has a different coding table.

TIdCoderMD2 (MD4 and MD5)

These are variations of the MD (Message Digest) algorithm for encoding. They are all hash algorithms; thus it is a one-way encoding system. There are no decoders.

Downloading Updated Internet Direct Components

In addition to finding Internet Direct components on Kylix's Component Palette, you can download them from the Indy Web site at http://www.nevrona.com/Indy/. The Indy Web site contains the latest version of Internet Direct components available as well as older versions, articles, and updated documentation. You will want to check this site periodically for updates and additions to the Internet Direct component set. Refer to the online help in the later versions for information on using updated or additional components.

To install Internet Direct components downloaded from the Web, you should uninstall the current Internet Direct components first and then reboot your computer. Note, however, that if you use the Internet Direct components downloaded from the Web, the appearance and order of components on the pages of the Component Palette differ from those shown earlier in this chapter. The Internet Direct components that shipped with Kylix (and Delphi 6) were required to display properly in 256-color modes. The ones distributed with the versions on the Web site are known as "Cool Icons." They appear considerably different.

At the time of this writing, the Indy Team is working on version 8.1. This version, as well as later versions that follow, include updated components as well as entirely new component classes. The discussion of Internet Direct features provided in this chapter and the following chapter assume that you are working with version 8.0. If you download a later version, please refer to the readme and/or other support files for information on components and features not currently available in version 8.0.

Using the Internet Direct Components

The components of Internet Direct can be categorized into general categories. For example, a large number of components are associated with building socket-based servers, while another set are associated with building socket clients. Each of these general categories is described in the following sections, along with how you might use them in your Kylix applications.

How Internet Direct Clients Work

Internet Direct employs blocking socket calls. This makes Internet clients easy to create. The client component attempts to connect to a server and then waits for the connection to complete. If the connection cannot be made, an exception is raised.

Blocking socket calls mean that all requests a client makes of a server need to be processed successfully in order to return information to the client. If one of the instructions fails, then nothing is returned to the client except the raised exception. This approach is similar to what you are accustomed to if you have ever used Pascal to read from a file. After opening a file handle, you attempt to read. If the operation succeeds, the function returns the value read; otherwise, an exception is raised.

When working with Internet Direct clients, you typically need to perform the following tasks:

1. Request a connection to the server.
2. Make a request of the server and read the server's response. Depending on the server, you perform this step once or repeat it many times. This constitutes the bulk of the customization you do to code for Internet Direct client components.
3. Terminate the connection with the server and disconnect.

Internet Direct is designed to provide a very high level of abstraction. Intricacies and details of the TCP/IP stack are hidden from you, the programmer, so that you need only be concerned with the tasks at hand.

Internet Direct clients allow you to write your socket calls in a sequential order. Following is a short example of an Internet Direct session involving an Indy client:

```
with IndyClient do begin
  Host := 'zip.pbe.com'; // Host to call
  Port := 6000; // Port to call the server on
  Connect;
  Try
    // Your code goes here
  finally
    Disconnect;
  end;
end;
```

This pseudocode segment begins by identifying the server to connect to, the computer on which the server is running (Host), and the port on which it is listening. Next, the client attempts to connect to the server. If the server's listener accepts the connection, the code in the `try` block executes, with the `finally` block ensuring

that you formally terminate the client connection. If the server's listener does not accept the connection, the call to Connect will time out and an exception is raised, displaying an error message to the user. If the connection is refused, there is no need to do anything further. In most cases, the client's responsibilities are this simple.

In Chapter 21, for instance, you will be working with an example of a ZIP code lookup client and server. (The ZIP code refers to the five-digit number used by the U.S. Postal Service to help address mail.) In this example, the ZIP code lookup client requests a connection from the ZIP code lookup server. If the request is accepted, the client then sends a ZIP code to the server, which should return the name of the city associated with that ZIP code. If the server fails to do this for some reason, an exception is raised and the client disconnects. If the server succeeds, the client receives the city name and then disconnects.

Using TIdAntiFreeze

Indy has a special component that transparently solves the user interface freeze problem sometimes associated with blocking sockets described earlier in this chapter. The presence of a single TIdAntiFreeze instance in an application allows the use of blocking calls in the main thread without interfering with the user interface.

The TIdAntiFreeze component works by internally timing out calls to the TCP/IP stack and allowing messages to be processed during the timeout periods. The external calls to Internet Direct components continue to block, meaning that your code works the same whether or not TIdAntiFreeze is present.

Since the user interface freeze is only affected by blocking calls in the main thread, TIdAntiFreeze only affects Internet Direct calls made from the main thread. If an application uses Internet Direct components in threads, TIdAntiFreeze is not required.

You should be aware that the use of TIdAntiFreeze slows the socket communications somewhat, since it must periodically interrupt to allow the main thread to process messages. Because of this, care must be taken to not allow too much time to be consumed in events caused by messages. These include most user interface events, including OnClick, OnPaint, OnResize, and many others. You can control this to some extent through the configurable properties of TIdAntiFreeze. Furthermore, with Internet Direct the use of TIdAntiFreeze is optional, giving you complete control over this issue.

How Internet Direct Servers Work

Internet Direct has a variety of server models. Which one you use depends on your needs and the protocol used. The next few sections provide an overview of the base Internet Direct server components: TIdTCPServer, TIdSimpleServer, and TIdUDPServer.

These three server components implement the three primary mechanisms for server access in Internet Direct. For example, the approach employed by TIdTCPServer is also the same as for any TCP-based protocol server. This includes the majority of the protocol servers. TIdSimpleServer is used internally by components such as TIdFTP (the FTP Client) to handle data connections. Finally, TIdTrivialFTPServer makes use of the technology employed by TIdUPDServer.

Overview of TIdTCPServer

TIdTCPServer is the most practical Internet Direct server component. TIdTCPServer creates a secondary listening thread that is independent from the main thread of the program. The listener thread listens for incoming requests from potential clients. For each client that it answers (accepts the connection request), it creates a new thread to specifically service that individual client connection. The events appropriate to that particular protocol are then fired within the context of that thread. A diagram of this relationship is shown in Figure 20-1.

A thread on the server listens for client requests, utilizing a new dedicated thread for each accepted request.

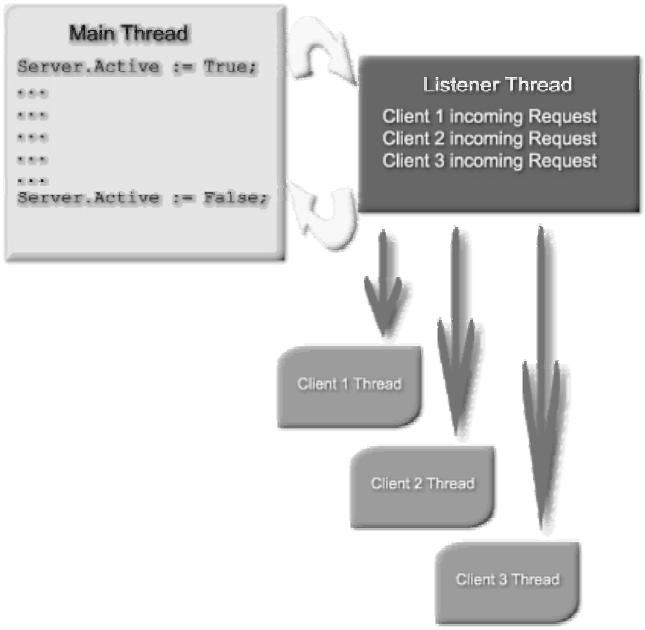


Figure 20-1 *Operation of TIdTCPServer*

Overview of TIdSimpleServer

TIdSimpleServer is used for creating single-use servers. A single-use server is one that can service only one connection at a time. Unlike TIdTCPServer, TIdSimpleServer does not spawn any listener or secondary threads. In other words, all of TIdSimpleServer's functionality is implemented within a single thread. Since all requests are handled by their own dedicated thread, client requests are serialized. As a result, if one client requests a connection while the server is communicating with another client, the request will be blocked until the current connection is disconnected, or the time-out period elapses, in which case the requesting client will fail to connect.

The TIdFTP client component utilizes TIdSimpleServer. When FTP performs a file transfer, a secondary TCP connection is opened to transfer the data and closed when the data has been transferred. This connection, called the "data channel," is unique for each file transfer. This approach allows the command channel to remain active and separate from the data channel, which is used only to transfer the files. The data channel is also binary, whereas the command channel is text. While the approach used by TIdSimpleServer is much simpler to implement than that used by TIdTCPServer, it is also limited. As a result, TIdSimpleServer is often used in clients that need to temporarily reverse their role and act as a server in order to process a special request. You typically do not use TIdSimpleServer for implementing servers in general.

Overview of TIdUDPServer

Since UDP is connectionless (each packet sent is "on its own" and is not part of a larger session or connection), TIdUDPServer does not require the complexity of TIdTCPServer. Instead, TIdUDPClient makes use of single-use listening methods. For example, while TIdTCPServer spawns separate threads for each connection, TUDPServer uses either the calling thread or a single secondary thread that handles all UDP requests.

When active, the TIdUDPServer creates a listening thread to listen for inbound UDP packets. For each UDP packet received, TIdUDPServer will fire the OnUDPRead event either in the main thread or in the context of the listening thread, depending on the value of the ThreadedEvent property.

When ThreadedEvent is False, the OnUDPRead event is fired in the context of the main program thread. When ThreadedEvent is True, the OnUDPRead event is fired in the context of the listener thread.

Whether ThreadedEvent is True or False, its execution will block the server from receiving more messages. Because of this, the any code that is inserted in the OnUDPRead event should be quick to process.

Using the Thread Manager

In Internet Direct version 8.0, TIdTCPServer is the only component that utilizes the thread managers. In future versions of Internet Direct, other components may utilize the thread management components as well. Because of this, thread management is encapsulated into separate thread management components. This also allows for multiple thread management schemes to exist, and new ones to be created.

TIdTCPServer has a ThreadMgr property and defaults to `nil`. If ThreadMgr is `nil` when TIdTCPServer is activated a TIdThreadMgrDeafault will be created implicitly. If ThreadMgr is set to a thread manager component, that component will be used for thread management instead.

Internet Direct Licensing

As mentioned previously, Internet Direct is open source and it is necessary to be aware of and follow the licensing policies if you want to use Internet Direct components in your applications. Internet Direct supports two different licensing policies. These include an Indy modified BSD (Berkeley Software Distribution) license and an Indy MPL (Mozilla Public License) license. These two licensing schemes provide you with flexibility. You should review which license better suits your needs, and use that license. If at a later time you decide to use the other license, you can change.

Indy Modified BSD License

The Indy Modified BSD license is a no-nonsense license that allows you to do almost anything you want with Internet Direct, provided you give proper attribution. The Indy Modified BSD License has been altered during Internet Direct's lifetime to be more flexible and may be changed again in the future. The changes cannot become more restrictive, only less so. Thus, it is important to check the Indy Web site for current licensing information. However, at the time of writing the Indy Modified BSD License is as follows.

Copyright

The following copyright statement must appear in your application:

Portions of this software are Copyright (c) 1993 - 2001, Chad Z. Hower (Kudzu) and the Indy Pit Crew - http://www.nevrona.com/Indy/.

License

The following licensing information must appear in your application:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation, about box and/or other materials provided with the distribution.

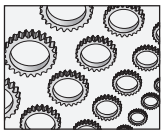
No personal names or organizations names associated with the Indy project may be used to endorse or promote products derived from this software without specific prior written permission of the specific individual or organization.

THIS SOFTWARE IS PROVIDED BY Chad Z. Hower (Kudzu) and the Indy Pit Crew "AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Indy MPL (Mozilla Public License) License

The Indy MPL (Mozilla Public License) license follows project Jedi's implementation of the MPL and allows developers to use its code in their applications ("Larger Work") regardless of whether the intended distribution will be in the public domain or in commercial applications, as long as the license conditions are met. Mozilla is the Open Source initiative formulated by Netscape for the next generation of their Web browsers. Netscape states, "We believe this license satisfies the Debian Free Software Guidelines which provide a commonly accepted definition of 'free software,' much like other free software licenses such as GPL (GNU Public License) or BSD."

Project Jedi projects typically have an MPL license. Internet Direct also has an MPL license v1.1 (see http://www.mozilla.org/MPL/MPL-1.1.html), to make it easier and consistent for Jedi users.



NOTE

Project Jedi is an international community of Delphi developers who develop and share Delphi code, components, and specialized technology in the public domain. Visit http://www.delphi-jedi.org for more information.

The MPL is much larger than the Indy Modified BSD License. Because of this, it is not included here. For a copy of the MPL license, including an annotated version, please consult the Indy licensing Web page at http://www.nevrona.com/indy/license.html.

Complying with Indy Licensing in Your Kylix Applications

When you use Internet Direct components in your Kylix applications, here is essentially what you need to do to comply with the open source Indy licensing policies.

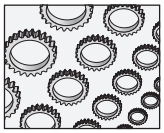
1. Visit http://www.nevrona.com/indy/license.html to obtain the latest versions of the two licenses: the Indy Modified BSD license and the Indy MPL license. This gives you more current versions of the licenses than those that ship with Kylix.
2. Choose one of the licenses.
3. Make sure you understand and comply with the directions given in the license and place the license text in comments in your source code.

Technical Support

Free support is available using the following resources:

- **Borland newsgroups** For free technical support from Team Indy, see borland.public.kylix.internet.web, borland.public.kylix.internet.sockets, borland.public.delphi.internet.winsock, or borland.public.cppbuilder.internet. These newsgroups are on the newsgroups.borland.com news server at port 119. Team Indy monitors these groups regularly, and everyone will benefit from your post.

- **Problem/Bug reporting** You can report bugs using the Bug Report form at: http://www.nevrona.com/indy/bugs/. Paid support is available from the following third party:
- **Indy Pro Priority Technical Support** If you are interested in professional, timely technical support for Internet Direct, Nevrona Designs provides several annual subscription plans to help you get over the bumps and pitfalls of Internet programming.



NOTE

These third parties are independent of the Indy Pit Crew, although several members of the Indy Pit Crew are part of this effort.

The next chapter focuses on using the Internet Direct components in your Internet applications.