

Remote Project54 application messaging via the Proxy Application

W. Thomas Miller, III

The Project54 proxy application (proxy.dll) supports version 2 messaging with remote applications (or remote application managers) via UDP/IP network packets. The proxy application listens continuously for incoming UDP/IP packets at a specific network address. Incoming UDP/IP packets from remote applications are converted by the proxy application into version 2 messages which are then submitted to the application manager. Outgoing version 2 messages delivered to the proxy application by the application manager are converted by the proxy application into UDP/IP packets and sent to the appropriate remote applications. While the proxy application was designed mainly to handle messaging between isolated remote applications (on a PDA for example) and a single application manager in the cruiser PC, two or more instances of the application manager running on separate computers can also be hooked together via proxy applications local to each instance.

Proxy Application UDP/IP Network Address

The proxy application listens for incoming UDP/IP packets at the IP address and port number specified in the \Project54\Proxy\Network registry path. The two relevant registry parameters are "ProxyIp" and "ProxyPort". The ProxyPort parameter specifies the IP port number (e.g. 3054) used by the proxy application to listen for incoming packets. ProxyIp specifies the IP address used by the proxy application to listen for incoming packets. If this address is "0.0.0.0" or if the parameter is omitted, the application accepts incoming packets on any of the local system's IP network adapters. If a specific IP address is given, the application only accepts incoming packets from the corresponding network adapter.

UDP/IP Message Format

A version 2 message has a source application name, a destination application name, a message identifier string, and a message body string. All four items are textual in nature. The format of a version 2 compatible message in a UDP/IP packet payload is as follows:

<P54V2>source;destination;identifier;messagebody

The UDP/IP packet payload above can be encoded as either 8 bit ASCII characters or as 16 bit wide characters. Native version 2 messages are 16 bit wide characters.

The source application name, the destination application name, and the message identifier string can not contain the character ';' (semicolon), since this is used as the field separator. The message body can contain any character other than the null character ((char)0 or (wchar_t)0).

For example, an incoming (relative to the local system) UDP/IP packet with the specific payload:

```
<P54V2>pdalights;lights;anyidentifier;STROBES OFF
```

causes the “STROBES OFF” message to be sent to the local “lights” application from the remote “pdalights” application.

As another example, an outgoing (relative to the local system) UDP/IP packet with the specific payload:

```
<P54V2>lights;pdalights;anyidentifier;STATUS STROBES OFF
```

indicates that the “STATUS STROBES OFF” message is being sent to the remote “pdalights” application from the local “lights” application.

When a UDP/IP packet with the above format is received by the proxy application, it is converted into a version 2 message by first checking the textual header (“<P54V2>”) and then parsing the four information fields, converting them if necessary to wide character format. The proxy application then checks the source name to see if this message is from a known (previously seen) application. If the message is from an unknown application, the proxy application adds the source name to its list of known applications and stores the IP address and port number that the message was received from, as well as the character size (8 or 16 bits), so that it can send outgoing messages to that application using the correct network address and using the remote application’s desired format. If the source application was previously unknown, the proxy application also sends a special message (“REGISTER APPLICATION ALIAS”) to the application manager. In response to this message, the application manager creates a new entry in its application list which contains the name of the source application matched to the interface pointer for the proxy application. *This requires version 2.1 or greater of the application manager.* Thus, any future messages on the local system which have the remote application name as the destination will be directed to the proxy application. Finally, the decoded version 2 message itself is submitted to the application manager.

When the proxy application receives a message from the application manager it first checks the destination name against its list of known applications. If a match is found (the destination application has been seen previously), the message is converted to the UDP/IP message format shown above (using 8 or 16 bit characters) and this packet is delivered to the network. If the name is not known (this should not happen normally) the message is ignored.

Known Applications

As described above, remote applications normally become “known” to the local proxy application, and thus to the local application manager, when a UDP/IP message is first

received from the remote application. However, the proxy application can also have a local list of known applications in the \Project54\Proxy\Applist registry path. Each parameter in this registry path has a name which is the name of the “known” application. The registry string value has the format:

ip.ip.ip.ip,port,f

where “ip.ip.ip.ip” is the IP address for the remote application, “port” is the IP port number used by the remote application, and ‘f’ is either ‘c’ for 8 bit character message format or ‘w’ for 16 bit format characters. For example, the registry entry:

pdalights REG_SZ 11.1.2.3,2054,c

indicates that the application named “pdalights” listens for messages on IP port number 2054 at IP address 11.1.2.3, and prefers 8 bit character message format.

Since the proxy application “learns” about remote applications automatically, it is not generally necessary to create this type of entry in the registry unless the local system needs to send messages to a remote application before any messages are received from that remote application.