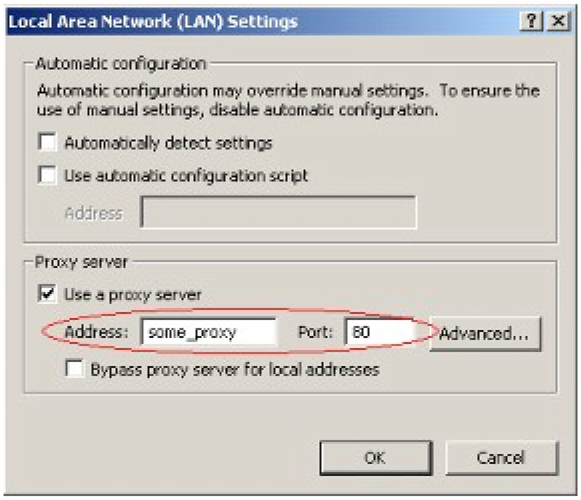
# HTTP proxy you need to bypass

Enter your proxy address and port here. “Your proxy” is the one that you use for surfing, and which actually blocks you from the Internet. If you don’t know what your proxy is, examine your browser settings.

## How to find your proxy **(please carefully read all cases):**

### 1. Fixed HTTP proxy:

If you use Internet Explorer go to “Tools (menu) – Internet options (item) – Connections (tab) – LAN settings (button)”. The following dialog window should appear:



Local Area Network (LAN) Settings

Automatic configuration

Automatic configuration may override manual settings. To ensure the use of manual settings, disable automatic configuration.

☐ Automatically detect settings

☐ Use automatic configuration script

Address

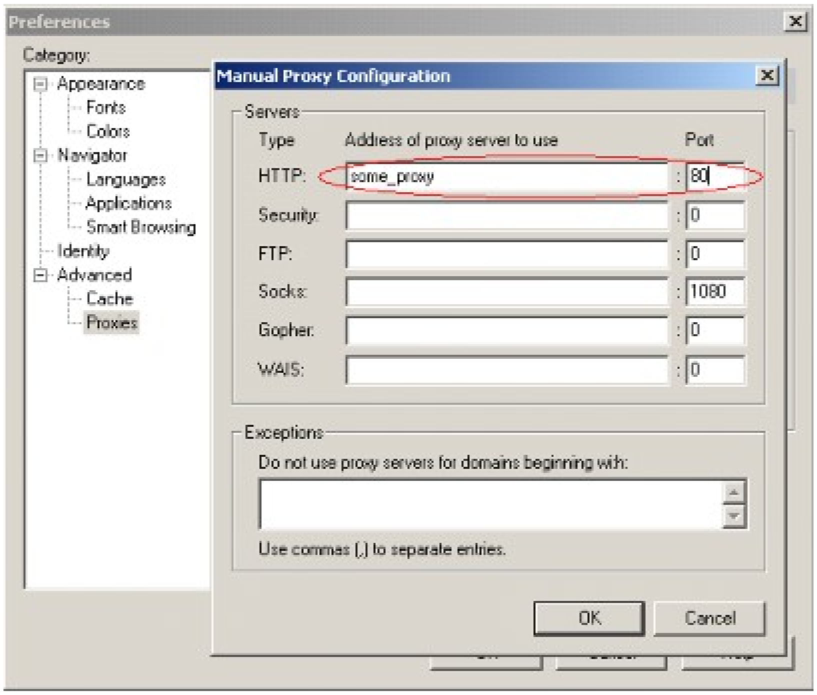
Proxy server

☑ Use a proxy server

Address: some_proxy Port: 80

☐ Bypass proxy server for local addresses

If you use Netscape Navigator go to “Edit (menu) – Preferences (item) – Advanced/Proxies (node tree at the left) – Manual Proxy Configuration View (button). The following dialog window should appear:



Preferences

Category:

- Appearance
  - Fonts
  - Colors
- Navigator
  - Languages
  - Applications
  - Smart Browsing
- Identity
- Advanced
  - Cache
  - Proxies

Manual Proxy Configuration

Servers

| Type | Address of proxy server to use | Port |
|---|---|---|
| HTTP: | some_proxy | 80 |
| Security: | | 0 |
| FTP: | | 0 |
| Socks: | | 1080 |
| Gopher: | | 0 |
| WAIS: | | 0 |

Exceptions

Do not use proxy servers for domains beginning with:

Use commas (,) to separate entries.

Now copy the proxy parameters that you have discovered into the HTTPort fields.

### 2. HTTP proxy assigned with scripts:

If your browser is set up to use automatic configuration script, rather than a fixed proxy setting, finding out a real proxy address could be tricky. For instance the configuration script your browser uses is http://some_host/proxy.pac

Try downloading the script file by pasting it's name (http://some_host/proxy.pac) into browser's line. If you examine the contents of the downloaded file, it may look like:

```
If (…)
    Proxy = host_1:port_1
ElseIf (…)
    Proxy = host_2:port_2
Else
    Proxy = host_3:port_3
EndIf
```

All host_X:port_X are actually your proxies. A different proxy could be activated depending on the logical expressions in brackets.

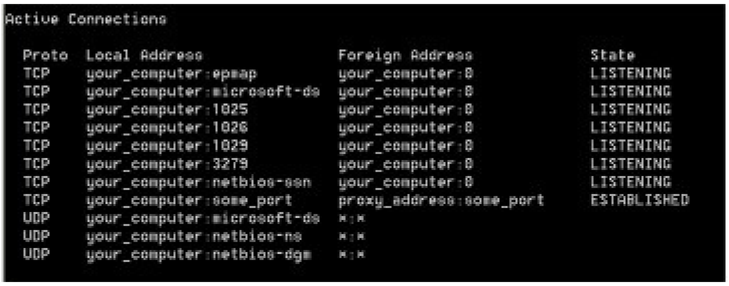
Now try entering host_X as your proxy host and port_X as yout proxy port. You may need to try all of the addresses you find in the configuration script. At least one of them should work.

### 3. HTTP proxy assigned with scripts (alternative method):

With your original browser settings, start downloading a big file from anywhere. While it loads, execute the following command from your command prompt:

```
Netstat /a
```

The typical output from netstat looks like:



```
Active Connections

  Proto  Local Address              Foreign Address            State
  TCP    your_computer:epmap        your_computer:0            LISTENING
  TCP    your_computer:microsoft-ds your_computer:0            LISTENING
  TCP    your_computer:1025         your_computer:0            LISTENING
  TCP    your_computer:1026         your_computer:0            LISTENING
  TCP    your_computer:1029         your_computer:0            LISTENING
  TCP    your_computer:3279         your_computer:0            LISTENING
  TCP    your_computer:netbios-ssn  your_computer:0            LISTENING
  TCP    your_computer:some_port    proxy_address:some_port    ESTABLISHED
  UDP    your_computer:microsoft-ds *:*
  UDP    your_computer:netbios-ns   *:*
  UDP    your_computer:netbios-dgm  *:*
```

See the ESTABLISHED connections in the list. There could be more than one established connection. One of them is your proxy. If there is a lot of established connections, you should try all hosts. For instance with the screenshot above you should enter proxy_address as your proxy host in HTTPort. Do NOT enter some_port as your proxy port, this is wrong. Some_port is just a random port, assigned to you by proxy.

Now, you know the proxy host, but how do you find out your proxy port ? Try the following frequently used ports: 80, 8080, 3128. If none of them work, download a portscan utility and use it on your proxy host to see what ports are open there. One of the open ports is the one you need.

### 4. No proxy at all, open access to any site (firewall or NAT).

If there is no proxy setting in your browser at all, you must be having an open access to all web sites. This is commonly done by your company firewall letting any traffic on port 80 out. Visit HTTPort site, check the FAQ section for the proxies lists. Pick any proxy (preferrably working on port 80) you like from the lists of available external proxies. Enter it in HTTPort as a proxy you want to bypass. This does not make much sense unless you realize that HTTPort needs an HTTP proxy. Just any proxy.

Note, that you should pick the fastest proxy, and if possible with CONNECT method support. The sites that maintain those huge proxy lists often provide you with the tools to check proxy speed and CONNECT method support.

**5. Transparent web accelerator (no proxy is specified in your browser).**

This one is sort of a proxy, but you don't know it's there. It hijacks all the requests that your browser issues and proxies them transparently (hence the name). The machine running transparent web accelerator can normally be used as a proxy, when asked directly. Therefore if you know the address of that machine you can enter it as a "proxy you want to bypass". Finding it can be tricky though. Try running "telnet www.yahoo.com 80" then type some garbage (may not appear on the screen) and press Enter twice. With web accelerator you should see "Bad request" answer, and it's normally accelerator that is answering, not the yahoo server. Therefore, in that result page look for any clues as to where the machine is. Something like "Generated [DATE] by [address] (proxy version)" or "MISS from [address]". There would be your address. In case you can't figure out the address of the accelerator, you can use any other web proxy as described in 4, although it could be a little slower.

With cases 4 and 5, as an option, you can use HTTHost itself as a proxy server. For instance, if you are running HTTHost personal edition at home (home.good.provider.com:80) and at work you are behind a firewall, not a proxy, you may set both “Proxy you want to bypass” and “Use personal host at” fields to home.good.provider.com:80. This way you will eliminate a proxy link from a chain, which speeds things up just a bit.

**Note:**

Do not enter 127.0.0.1 as a proxy that you want to bypass, this is (normally) wrong.

# Proxy authorization options

You need to set this only if the password is required to surf through your proxy. If you are being prompted for a username and password every time you start surfing through your proxy, you should enter these username and password here.

Check the “Authorize” checkbox and type in your username and password in the fields below.

Note: if you have to enter DOMAIN, username and password to authenticate with the proxy, you must enter username in HTTPort as DOMAIN\username

**New in version 3.SN:**

Now HTTPort should support different authentication schemes, not only Basic. Basic is the default and if it fails, HTTPort switches to the different mode of operation, in which it does not actually care for any authentication. It relies on Windows HTTP authentication handling to do all authentication.

**Recommended initial setting: Your proxy username and password if you have it.**

## “User-Agent” option

In its original meaning, user-agent is any client program that uses HTTP protocol. HTTP is a client-server request-response protocol. Every time the client sends a request to the server, it introduces itself to it, by adding a special “User-Agent” field to the request. Most of HTTP servers (and proxies) do not make any use of this field (except for collecting statistics), but they can.

The User-Agent dropdown list allows you to select one of the most popular browsers, so that every time HTTPort makes a request through proxy, the proxy thinks that the request was sent by some browser, not by HTTPort. It’s recommended that you set this parameter to the browser that you actually use.

If your proxy is filtering User-Agents, allowing for instance only “User-Agent: Super IE Patched For Maximum Overkill” through, here is what you should do. Obviously there is no matching User-Agent in the dropdown. Set the “User-Agent” dropdown box to “None”, and create httport.hdr file in the directory where HTTPort is installed. This file may contain any arbitrary lines which will be added to every request HTTPort sends to proxy. Therefore, create an httport.hdr file with a single line:

User-Agent: Super IE Patched For Maximum Overkill

So your proxy will see HTTPort as the patched IE now and let it through.

**Recommended initial setting: The browser you use for surfing.**

## “Bypass mode” option

This can be set to either “Auto”, “SSL/CONNECT” or “Remote host”.

It’s recommended that you use “Auto”, unless you need the features available in “Remote host” mode only, such as full SOCKS support and data encryption.

“SSL/CONNECT” mode uses HTTP CONNECT method to establish a virtual tunnel through a proxy. Not all proxies support this method, so this mode is not guaranteed to work. This is the fastest mode, and it’s recommended that you use it if your proxy allows that.
This is a standalone mode, in which HTTPort does not require HTTHost support.

“Remote host” mode uses a special server software (HTTHost) which is running on a server outside of a proxy. HTTPort automatically locates available HTTHost and establishes a link to it through the proxy. In this mode, HTTPort sends a series of HTTP GET requests to the HTTHost, which instruct the HTTHost what operations to perform – where to connect, what data to send etc. So, briefly, HTTHost is a remotely controlled TCP/IP unit. This mode is pretty slow, for lots of reasons, that will not be mentioned here. The inbound throughput of HTTHost – HTTPort link is limited with around 20-30K/sec at most.

“Auto” mode is just what it says. It tries to use “SSL/CONNECT” first, and if it fails (the proxy does not support CONNECT method) it switches to “Remote host” mode. So, if you see “Auto” being used, this means in fact that “SSL/CONNECT” is being used, otherwise the mode would have been chaged to “Remote host”.

**Recommended initial setting: Auto.**

## “Start/Stop” button

This is a main switch. Clicking “Start” brings HTTPort to work, clicking “Stop” terminates all activity. HTTPort won’t do anything when not started. All the settings are disabled when HTTPort is started, so the recommended usage scenario is: think, adjust settings, start. Once something’s wrong: stop, think, adjust settings.

HTTPort will start automatically if the “Start and minimize” checkbox is checked.

## Use personal remote host at

If you use “Remote host” mode, HTTPort makes use of external server software (HTTHost). By default HTTPort uses one of the publicly available HTTHosts automatically. Therefore in most cases you should not enter anything in these fields (it is recommended that you leave it blank).

**Recommended initial setting: blank.**

If you leave this field blank (which is recommended), HTTPort will use any publicly available HTTHost. This is perfectly ok, those public HTTHosts run 24 hours a day 7 days a week and they are fast.

There were users though that required a possibility that they run their own HTTHost on their own PCs outside of a proxy. Just for instance they wanted to run HTTHost at home, so that they use their own small HTTHost to connect HTTPort to it from behind a proxy at work. This option is for them.

You should use this option only if you have a capability of installing an HTTHost on some PC outside of a proxy. If you have downloaded personal HTTHost edition from HTTPort site and installed it at home.server.good.provider.com:80, you should enter home.server.good.provider.com and 80 in these fields. From this point HTTPort will use your personal HTTHost only. It’s up to you to feed it and care of it.

The password value must match with the “Personal password” parameter in your personal HTTHost.

Pro mode on.

Note: it’s recommended that you set your personal HTTHost for listening port 80, therefore the port setting here in HTTPort will be 80 as well. If you have something (web server ?) already running on port 80 on the PC you want to install personal HTTHost at, you’d better switch the web server to other port, for example to port 81, and use the HTTHost passthrough feature to share with 127.0.0.1:81. This way both HTTHost and web server will be working on port 80 (or at least it will look this way from outside).

If your personal HTTHost is running on port other than 80, you may have problems, because
1. proxy may block requests to ports other than 80.
2. HTTPort NTLM authentication may not work.

Pro mode off.

This option is only used in “Remote host” mode.

## “Start and minimize automatically” option

This checkbox, when checked, makes HTTPort starting every time it’s run. That is, you don’t have to click Start button manually after you have run HTTPort.

Use this checkbox when you are satisfied with all other settings. Otherwise you will have to stop HTTPort to make any changes and then start it again manually using Start/Stop button.

**Recommended initial setting: not checked.**

## “Hide on minimize” option

This checkbox allows you to hide HTTPort. When this checkbox is not checked, HTTPort minimizes into tray. When you check this checkbox, HTTPort deletes it’s icon from the tray, so it’s nearly invisible. It cannot be switched to by “ALT+TAB”, and it does not visually appear on the screen. The only way too see it running is to use Windows NT/2000 Task Manager.

To un-hide HTTPort, press Ctrl+Alt+H simultaneously.

**Recommended initial setting: not checked.**

## “Accept only connections from this PC” option

This checkbox, when checked, will make HTTPort invisible for port scanning.

For instance you have HTTPort mapping local port A to some remote server and port. When this checkbox is clear, anyone from your surrounding network can connect to your PC at port A. In case they use portscan on your PC, they see that your PC has port A listening. This may result in questions, which is not a good thing.

It’s recommended that you check this checkbox. In this case all HTTPort listening activity wil be bound to localhost (127.0.0.1) only. This means that the connection will only be accepted by HTTPort if it originates from localhost, i.e. from the same PC HTTPort is running at. No external connections will be allowed and no portscanning can determine the presence of HTTPort.

**Recommended initial setting: checked.**

# “Display remote host errors” option

When checked, this checkbox will make errors, reported by remote host(s) to appear in the Errors log.

This only makes sense in “Remote host” mode and it’s for debugging purposes only. It’s not recommended that you check this option unless you really need it.

Note: if you see remote host errors, do not panic. You do not know what these errors mean, right ? Just for instance, you should ignore the following remote host error messages

0103 Invalid connection id
0202 Connection state closed
0203 Connection state disconnected

Once again: this is for debugging only.

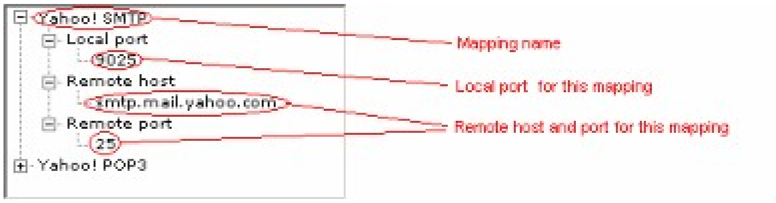
This setting is not saved on exit.

**Recommended initial setting: not checked.**

## Port mapping

Understanding this is a key to making HTTPort working.

The following screenshot shows a mapping window with two mappings defined. First one is expanded to fully show it's parameters, another is collapsed.



⊟ Yahoo! SMTP
⊟ Local port
9025
⊟ Remote host
smtp.mail.yahoo.com
⊟ Remote port
25
⊞ Yahoo! POP3

Mapping name

Local port for this mapping

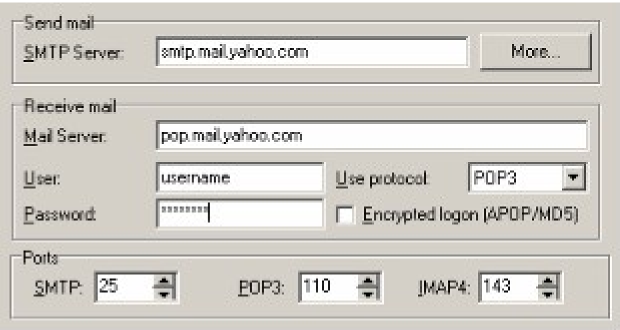
Remote host and port for this mapping

What HTTPort does is that it helps all the local TCP/IP applications to connect to the servers they need.

For instance the application, such as mail client, might want to connect directly to smtp.mail.yahoo.com:25 which is an SMTP server for Yahoo!, but the proxy will not allow that.

The mail client can still connect to localhost, i.e. to HTTPort, which will run the connection through a tunnel. This is the idea – you point your mail client to 127.0.0.1:9025 as an SMTP server for Yahoo! and in HTTPort map local port 9025 to the original server of smtp.mail.yahoo.com:25. HTTPorts intercepts the connection to local port of 9025, and transfers all the application data to and from the remote server. This is absolutely transparent and neither the application (mail client) nor the mail server know about that the connection is actually tunneled.

This is how the mail client could be set up if there is no proxy:

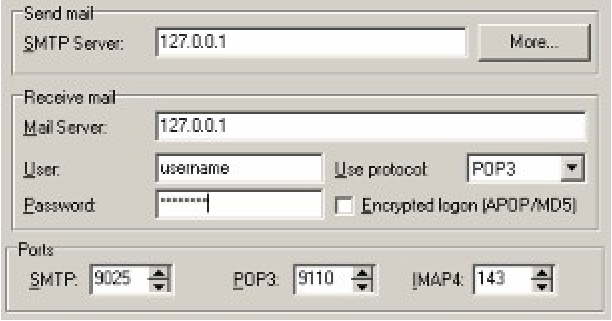


Send mail
SMTP Server: smtp.mail.yahoo.com

Receive mail
Mail Server: pop.mail.yahoo.com
User: username
Use protocol: POP3
Password: ********
□ Encrypted logon (APOP/MD5)

Ports
SMTP: 25
POP3: 110
IMAP4: 143

This is how it should be set up for use with the above HTTPort mappings:



Send mail
SMTP Server: 127.0.0.1

Receive mail
Mail Server: 127.0.0.1
User: username
Use protocol: POP3
Password: ********
□ Encrypted logon (APOP/MD5)

Ports
SMTP: 9025
POP3: 9110
IMAP4: 143

The idea is the same with any other application. In order to run some software through the HTTPort, you should:

1. Determine the address and port of the server your application wants to use. Examine your default application settings. Most probably it's what you need.
2. Create a new mapping in HTTPort that maps some local port (preferrably above 1024) to that remote server and port from step 1.
3. Point your application to 127.0.0.1:some_local_port as if it was the original server it needs.

**Notes:**

You need a separate mapping for every remote server.

All the mappings are independent from each other.

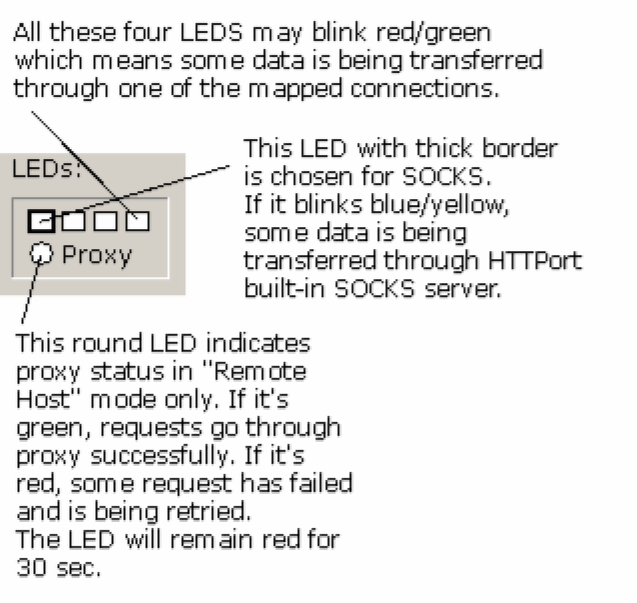
You cannot use the same local port for two mappings. Local ports cannot be shared. Once a local port is bound to one mapping, you can't use it with another. Pick another one from range 1024..65535.

To edit a mapping, make sure HTTPort is stopped and right-click the parameter you want to edit.

## Add and Remove buttons

Add button creates a new blank mapping. Remove button deletes the selected mapping.

# LEDs (Light Emitting Diodes)



All these four LEDS may blink red/green which means some data is being transferred through one of the mapped connections.

This LED with thick border is chosen for SOCKS. If it blinks blue/yellow, some data is being transferred through HTTPort built-in SOCKS server.

This round LED indicates proxy status in "Remote Host" mode only. If it's green, requests go through proxy successfully. If it's red, some request has failed and is being retried. The LED will remain red for 30 sec.

These LEDs are just an eye candy. Blinking means working. Not blinking means not working. When some LED is blinking red/green it means that data has been sent/received through one of the mappings.

The first LED (the one with the thick border) also reflects built-in SOCKS server activity. When it blinks yellow/blue, it means that data has been sent /received through the SOCKSified connection. This is the only way you can monitor the SOCKS activity in HTTPort. No detailed statistics is provided in current version.

LEDs only turn white when the connection is closed. Therefore if a LED is red/green/blue/yellow it does not mean that data is being sent/received at this very moment. It only shows the last connection state (send or receive).

**New in version 3.SN:**

The round LED marked “Proxy” indicates proxy status. This LED is only working in “Remote Host” mode. If it’s green, some requests did go through the proxy successfully. If it’s red, some request did fail and is now being retried. If you see it red, it does not necessarily mean that your proxy can not be bypassed. It only means that the proxy did fail to execute request. This failure may be temporary, and HTTPort is trying to recover from this failure. Once this LED has turned red, it will remain red for 30 seconds, no matter if some requests will succeed during this time. This is for you to notice that the proxy has actually failed.

## “Run SOCKS4 server (port 1080)” option

SOCKS is a protocol designed by NEC to allow software to tunnel TCP/IP through the SOCKS proxy. The difference between SOCKS and original HTTPort concept is that with original HTTPort you have a set of fixed predefined mappings, whereas with SOCKS you can dynamically allocate the mappings.

Obviously there is no problem with making HTTPort a SOCKS server, so here it is. HTTPort can now be used as a SOCKS server at local PC. If your application needs a SOCKS proxy, just check this checkbox and point the application to 127.0.0.1:1080 as a SOCKS server.

Built-in SOCKS support now makes it possible to use HTTPort with ICQ, FTP and other useful stuff.

Please note, that if this checkbox is checked, and the “Full SOCKS support” checkbox is not checked, only half of a SOCKS protocol is supported. Specifically – SOCKS.BIND method is not supported.

You can bind the built-in SOCKS server to any other local port other than 1080. It is not recommended, but you can do it, by editing the httport.ini file. Add a line SocksPort=X to the httport.ini file to set HTTPort SOCKS server to listening port X.

**Recommended initial setting: checked.**

## “Full SOCKS4 support (BIND)” option

This checkbox, when checked, makes HTTPort a fully compatible SOCKS4 server. SOCKS protocol defines two methods – SOCKS.CONNECT and SOCKS.BIND. SOCKS.BIND method is only available with “Remote host” mode and its support is only turned on when this checkbox is checked.

Note that ICQ attempts to use SOCKS.BIND at startup (probably for detecting whether this capability is present), but it does not actually use SOCKS.BIND when sending and receiving messages. Therefore if you leave this checkbox clear and attempt to run ICQ through HTTPort’s SOCKS server, HTTPort will report an error, but it does not mean that ICQ won’t work, on the contrary, ICQ works fine without SOCKS.BIND.

**Recommended initial setting: checked, if enabled.**

## “Enable data encryption” option

Please read carefully– this is very important.

This option is available only with “Remote host” mode. When this checkbox is checked, all traffic between HTTPort and HTTHost is encrypted using strong encryption algorithms.

**Recommended initial setting: not checked.**

There is a big issue about strong cryptography. **In certain countries you are not allowed to use strong cryptographic algorithms.** Therefore, if you reside in one of such a countries, for instance: Belarus, China, India, Israel, Kazakhstan, Mongolia, Pakistan, Russia, Saudi Arabia, Singapore, Tunisia, Venezuela, Vietnam, do NOT turn this feature on, or you may face troubles.

HTTPort uses RSA (1024 bit key size) and Blowfish (192 bit key size).

RSA algorithm is in public domain since September 6, 2000, and Blowfish is in public domain right from the very beginning, therefore there should be no patent problems.

HTTPort uses Blowfish and RMD-160 implementations from public domain DCPCrypt library. © David Barton (davebarton@bigfoot.com).

HTTPort uses its own RSA implementation (rkeyproc.dll).

When this feature is enabled, it’s impossible (well, almost impossible) to log your real activity. All the data and destination servers that you use are transferred in encrypted form, therefore it is almost impossible to understand what you are doing even if they have full access to the traffic. Once data leaves your PC, it’s protected. Once it gets to HTTHost, it’s being decrypted and goes plain the the target server.

**If you would like to use HTTHost personal edition, please be careful:** anybody can download this personal edition, and it goes with the same HTTHost key for everybody. Therefore you are not protected with mathematics, which is the only perfect protection. Anybody knows your HTTHost key, therefore anybody can decrypt your data.

Once again – personal edition of HTTHost DOES encrypt the data, but this encryption can theoretically be easily decrypted by anyone. In order to be perfectly safe, you should get your own private HTTHost key and use it with your personal edition of HTTHost. You can order your own you key from HTTPort site. Once you have your unique key, it’s up to you not to show it to anyone (i.e. keep it secret).

This is how it works:

You create your random Blowfish key:

1. You draw something on the pad.
2. This pad is actually a bit array. HTTPort calculates RMD-160 hash function of that array.
3. This hash data is a seed for a random number generator. HTTPort uses XORed outputs of regular C rand() and Mersenne Twister generator. rand() takes another 32 bits of seed (current time). I'm aware that these 32 bits are not random at all, so saying 192 bit is a little hype, you may think of just 160 bits.
4. 24 random bytes are generated and this becomes your key, denoted as Bk.

Key negotiation:

(Me, Md) is the master RSA key. Em and Dm is Master RSA encryption and decryption respectively.
(He, Hd) is a HTTHost RSA key. Eh and Dh is HTTHost RSA encryption and decryption respectively.

H is a hash function (derived from RMD-160).

1. All HTTPorts go with Md. All HTTHosts go with Hd and precalculated Em(He).
2. HTTPort sends H(Bk) to HTTHost.
3. HTTHost checks its key cache to see if the key with that hash is already known. If it is, go to 10.
4. If not, HTTHost responds with Em(He).
5. HTTPort calculates Dm(Em(He)) = He using Md.
6. HTTPort calculates Eh(Bk) using He and sends it to HTTHost.
7. HTTHost calculates Dh(Eh(Bk)) = Bk using Hd.
8. HTTHost responds with H(Bk).
9. HTTPort checks the hash to match. If it matches, both sides have the key.
10. Bk is used for generating a Blowfish key schedule.
11. From that point, every request that goes from HTTPort to HTTHost has an additional parameter H(Bk). Both HTTPort and HTTHost know the key schedule for key with that hash, and they both encrypt to that key schedule.

So, HTTPort selects a random key once, encrypts it with password and stores it locally. The first time HTTPort connects to HTTHost they go through key exchange procedure above. Once HTTHost knows Bk, it caches it under H(Bk), so that next time the check in step 3 will allow to avoid using RSA.

This approach is vulnerable to the replay attack, since the key remains constant for a long time. Therefore it is not recommended that you control the nuclear reactor in your basement using HTTPort. Nevertheless, this does not make it simpler to decrypt the data.

This is also vulnerable to key theft at HTTHost side, since HTTHost key and cached client key schedules are stored locally in plain.