ObjectValuE

Personal VPN Solution White Paper

Version 1.3, 14th October 2003

Client	N/A
Contract No.	N/A
Project / Proposal Name	Personal VPN Solution
Project / Proposal No.	Q030001
Document ID	20031014_PersonalVPN_v13.doc
Status	Issued
Prepared for ObjectValue Ltd.	Immo Hüneke
Reviewed	
Approved	
Authorised	
Agreed (Client)	
Distribution	

Summary Table of Contents

1 Executive Overview

1.1 Purpose

This white paper outlines a technical solution to a common problem, which is intended as the basis for a commercial proposal to mobile operators and directly to users in enterprises of any size.

1.2 Scope

The solution outlined has been built for a pilot implementation from readily available open-source and proprietary (AppGate) components. A few days of software development were required to augment the AppGate Server and PC client.

This solution is intended to be used by no more than a few thousand pilot users at a time. The mobile operator will need to procure an AppGate Server to host these pilot users. A small number of pilot users can be accommodated on AppGate's own demonstration server. Once the principle and cost-effectiveness of mobile access to corporate servers and applications has been established, the enterprise would be expected to set up and manage its own mobile gateway. For smaller enterprises, however, it might prove cost-effective to use a hosted remote-access service instead of investing in their own AppGate Server, and in such cases this solution might still apply.

1.3 Summary

There are clear economies to be achieved by equipping knowledge workers with mobile access to enterprise applications, such as e-mail, scheduling, contacts and intranet Web servers. Access to ERP, CRM and custom applications may follow. The return on investment (ROI) to be expected is most clearly demonstrated in a user trial lasting at least a few weeks. However, trialling such mobile access can involve a considerable investment in infrastructure and also require the enterprise concerned to modify its security policies, which puts considerable obstacles in the way of the technology trailblazers in any company. They cannot just go out and buy a mobile device and connect it to their enterprise network. Consequently the sale of mobile services to enterprise users is more difficult than it should be.

The solution outlined in this paper requires the triallist merely to download a small piece of software to their personal computer or workstation, and leave the machine running while out of the office. It doesn't conflict with any security policies, yet it works through enterprise firewalls and web proxy servers.

2 Document Control

2.1 Table of Contents

2.2 Amendment History

Version	Date	Changed by	Reason / Comment
1.1a	29th May 2003	Immo Hüneke	Initial version
1.2a	11th June 2003	Immo Hüneke	Revised to eliminate need for remote port forwarding
1.2b	13th June 2003	Immo Hüneke	Revised following review by AppGate
1.3	14th October 2003	Immo Hüneke	Revised following completion of the pilot software implementation

2.3 Change Forecast

None at present.

2.4 References

[APPGTW]	Wireless Applications in a Practical Mobile Computing Infrastructure, AppGate Network Security white paper, 10 January 2003, Jamie Bodley-Scott and Håkan Pernsved
[IDENT]	Identification Protocol, RFC1413, http://www.ietf.org/rfc/rfc1413.txt, February 1993, M. St. Johns
[OPNSSH]	OpenSSH, http://www.openssh.org/, release 3.6.1, 1 April 2003, Aaron Campbell, Bob Beck, Markus Friedl, Phil Hands, Damien Miller, Niels Provos, Theo de Raadt, Dug Song and many others
[SSH]	Secure Shell (secsh), http://www.ietf.org/html.charters/secsh-charter.html, 10 April 2003, Bill Sommerfeld et al

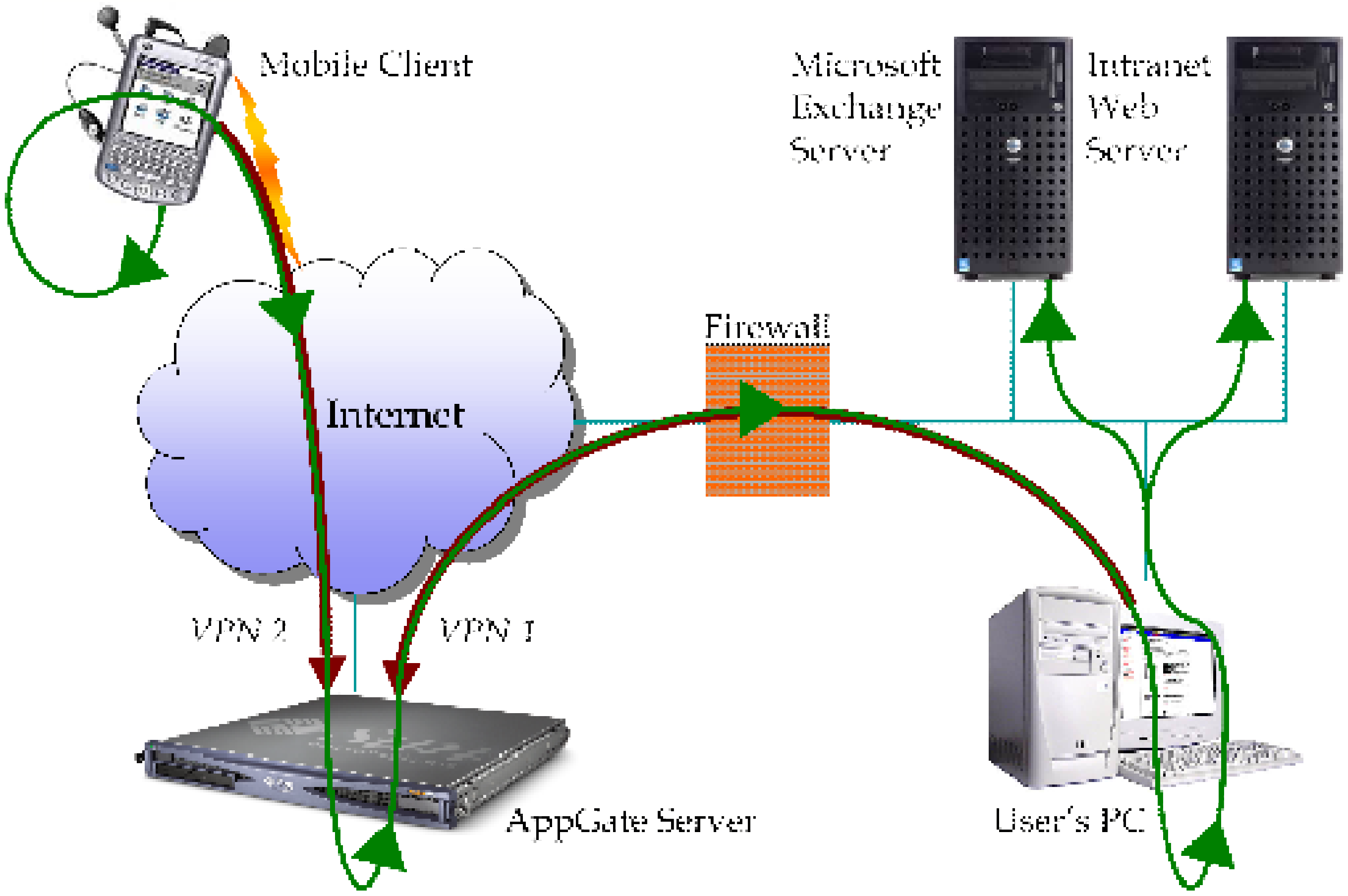
2.5 Glossary

Ident	The Identification Protocol defined by RFC1413 [IDENT] provides a way for a process to discover the identity of the user opening an IP socket connection to it. It can be thought of as an analogy to the Caller ID service of telephone networks. The information (if available) is provided by a daemon called **identd** running on the originating machine
IP	Internet Protocol (colloquially equivalent to TCP/IP, though other IP transports exist, such as UDP/IP)
Port	A sub-address for an IP address. A process listens for connection requests addressed to a particular port number. When a request arrives, the IP stack opens a session, which is uniquely identified by the port numbers and IP addresses of both its ends, and gives the listening process a reference to it (see socket)
Proxy	A Web proxy server is used both to improve network performance by caching the results of HTTP and FTP requests and to improve network security by isolating the client from the Web server. Web clients within an enterprise LAN access external Web servers via the proxy
Socket	A socket is a programming construct that provides a convenient abstraction for one endpoint of an IP session. Once opened, it is treated in most respects like a file that can be read and written
SSH	Secure Shell – ssh is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network [SSH], [OPNSSH]. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel
SSL	Secure Socket Layer – also known as TLS (Transport Layer Security). A protocol designed by Netscape Communications Corporation to provide encrypted communications on the Internet. SSL is layered above the connection protocol TCP/IP and beneath application protocols such as HTTP, resulting in the access method HTTPS

Tunnelling	Any IP protocol can be transported between two networks by tunneling via some other TCP/IP protocol. For example, the Squid proxy server can tunnel secure FTP and secure HTTP requests where a firewall disallows direct connections from the client to the server
VPN	Virtual Private Network – The use of encryption in the lower protocol layers to provide a secure connection through an otherwise insecure network, typically the Internet

3 Solution Description

The following illustration shows the essential elements of the solution. They are individually specified below.



3.1 AppGate Server

The AppGate Server is the lynchpin of the approach outlined in this paper. Both the mobile client and the workstation connect to this server using VPN connections secured using the SSH protocol. The VPN clients authenticate themselves to the AppGate Server by means of a username and password, or a more rigorous mechanism such as an RSA SecurID token.

The AppGate Server, as standard, provides a user-specific configuration to each VPN client, which includes a list of port numbers on the client to be used for port-forwarding. Each connection request from a program running on the client machine to one of these ports on the client machine will be relayed via the VPN to the specified remote server and port.

In order to implement the remote access solution, the client connection requests are forwarded to a specially-developed daemon process called **ag_patchd**, which runs on the AppGate Server itself and acts like a patch-panel for forwarded ports. Matching mobile client and office client sessions are "plugged together„ by **ag_patchd** to form a long tunnel from the mobile client to the office client. The daemon is also referred to as a *Rendezvous Server*.

3.1.1 **ag_patchd** details

There's a separate instance of **ag_patchd** for each protocol to be supported (HTTP, FTP, POP3, IMAP and so on). Most protocols use one conventional port number, but some use more than one. So, strictly speaking, there is an instance of **ag_patchd** for each port number. Each instance listens on two ports: one for office clients to connect to and one for mobile clients.

In order to match connection requests from office clients with those from mobile clients, **ag_patchd** must

- identify the type of client (office or mobile) – this can be deduced from the port number on which the connection was made
- identify the AppGate user – it does this by calling **ident_id** (from the C library **libident**) on the socket descriptor, which returns a user ID string derived from the user-name and password used to connect the AppGate Client to the AppGate server

The daemon maintains a table of open, waiting office client sockets, indexed by user ID. When a request comes in from a mobile client, it is matched to the first available office client session that has the same user ID, which is removed from the table. The two sockets are then handed to a worker thread (or a child process created using the **fork** system call) that transparently moves data from each socket to the other. As soon as either input socket is closed (indicated by a received data length of 0), the worker thread closes the other socket and returns to the thread pool (or exits, if it is a child process).

3.1.2 **ag_patchd** invocation

The daemon is started using a command line such as

```
ag_patchd -p HTTP -o 10080 -m 20080 -t 1800
```

where the switches have the following meanings:

-m Mobile client port number – the port number to which mobile clients connect

-o Office client port number – the port number to which office clients connect

-p Protocol – not currently used, but helpful in process listings to distinguish the instances

-t Time to live – if the process has been idle for this number of seconds, it terminates.

A perl script (**server.pl**) is supplied that can be configured to start any number of daemons and monitor them, immediately starting up another copy if any daemon exits.

3.2 Virtual Private Networks

The SSH protocol conventionally uses TCP port 22. VPN2 is established by connecting to an SSH server listening on this port on the AppGate Server.

However, VPN1 has to traverse a firewall. Many corporate firewalls block outgoing connection attempts to any TCP port except certain well-known ones, such as HTTP (port 80) or HTTPs (port 443). In other cases, no outgoing connections are permitted at all – every connection attempt is intercepted and handled by a proxy server.

Because HTTP and HTTPs protocols have to be supported by proxy servers or firewalls in order to allow users on the corporate network to browse the WorldWide Web, VPN1 connects to an SSH server listening on the HTTPs port on the AppGate Server. However, the protocol that runs across this connection is SSH, not SSL. This enables connections to be tunnelled through the VPN.

Many enterprise networks do not allow PC browsers to connect directly to external Web servers. The AppGate client can therefore be configured to connect to the AppGate server through a proxy server for HTTPs. It supports HTTP as well as SOCKS modes of operation, depending on the protocols supported by the proxy server in use.

3.3 Mobile Client

The e-mail client on the mobile device is configured to use “localhost„ as its mail server. The mail client’s connection requests are tunnelled by the AppGate Lite client to an **ag_patchd** process on the AppGate Server, which forwards them through the VPN1 tunnel to the office client and hence to the mail server.

The same principle can be extended to arbitrary further applications and TCP/IP protocols. In each case, the port-forwarding rules are communicated to the AppGate Lite client by the AppGate Server at the time of connection and user authentication, so there is no need to configure anything on the AppGate client.

Typically, the mobile web browser needs to connect to multiple web servers simultaneously or one after the other, so this simple scheme can’t be used. Instead, the web browser on the mobile client is configured to use a proxy server for all page accesses. The proxy server used is “localhost„. However, there isn’t actually a proxy server running on the mobile device: instead, the AppGate Lite client intercepts connections to localhost and tunnels them through VPN2 to an **ag_patchd** process on the AppGate Server. It in turn tunnels the connection request to the office client through VPN1 and hence to a proxy server running either somewhere on the office network or directly on the user’s PC (see below).

Whenever the user wishes to access the applications on the corporate network, if there is currently no VPN in place, the AppGate Lite client must be started and a valid username and password supplied to connect to the AppGate Server. It may be possible to automate the connection procedure to some extent. When the connection has been made, the AppGate Server downloads the user’s configuration to the mobile client. This configuration specifies all of the port numbers to be forwarded from localhost to an **ag_patchd** process on the AppGate Server, and the corresponding **ag_patchd** mobile client port numbers.

3.4 User’s PC – Office Client

The standard AppGate Client runs on the PC (or on a networked server). However, it is configured to connect to Port 443 (HTTPs) on the AppGate Server instead of the standard Port 22 and optionally configured to connect via a firewall proxy. The office client retrieves the same user configuration from the AppGate Server as the mobile client, including all of the supported port forwardings. However, because the AppGate Server supports client-device-specific configurations, each of the port numbers is different to that given to the same user’s mobile client.

Up to two additional processes run on the office client.

Firstly, the counterpart of **ag_patchd** is the **ag_patchd Office Client**. Its job is to open IP sessions (tunnelled through the AppGate Client) to **ag_patchd** for each port number configured. A configuration dialog allows the user to specify which localhost port numbers the Office Client connects to (these must match the port forwards specified in the user's AppGate configuration) and the intranet server and port number corresponding to each one. When data starts arriving down one of these sessions, the Office Client sets up a connection to the corresponding intranet server and initialises a worker thread to forward stream data transparently between the upstream and downstream sockets until the session is closed by either end. In parallel, the Office Client requests a new connection to the same port number on localhost, which will be forwarded to **ag_patchd** ready to receive the next connection request from the mobile client.

Secondly, a simple local Web proxy (such as the free AnalogX Proxy – see http://www.analogx.com/contents/download/network/proxy.htm) may run on the workstation, unless a suitable proxy server is available elsewhere within the office network. Each page access requested by the mobile client is forwarded to this proxy using the VPN tunnels. The proxy server then issues the HTTP or FTP request to the origin server (e.g. an intranet Web server) on behalf of the mobile Web browser.

3.5 Caveats

This scheme is designed merely to demonstrate the cost-effectiveness of mobile access to productivity applications such as e-mail, calendaring, contact information and internal Web services, including secure (HTTPs) pages. More sophisticated mobile applications, such as those requiring direct connections to databases, may be supported, but it cannot be guaranteed without further experiment.

4 Deploying and Managing the Solution

4.1 Download and Installation

Downloadable packages are provided under http://www.objectvalue.com/ for the daemon and office client portions. In each case, the installation procedure consists of uncompressing the downloaded archive to a suitable location. To uninstall the program, simply delete the folder created by the installation procedure.

4.1.1 Daemon Installation

4.1.1.1 Linux

Download both the binary and source RPMs. Log on as root and install using

```
rpm -Uvh ag_patch*rpm
```

This will put the binary into **/usr/local/bin** and the sources into **/usr/src/redhat/SOURCES**.

Within the sources, find the Perl scripts **server.pl** and **Server.pm** and copy these to **/usr/local/bin**, renaming **server.pl** to **ag_patchd_server.pl** to avoid any confusion. Edit the file to add as many daemon instances as you require to support the port forwards configured on the AppGate server.

You need to arrange to run **ag_patchd_server.pl** automatically when your machine starts up. There are a number of strategies for doing this, including placing symbolic links to the script into **/etc/init/rc*.d**, running a periodic watchdog under cron and listing the script at the end of **/etc/rc.sysinit**. How you do it depends on the system management policy of your server. The script takes no parameters (i.e. it will ignore any that it does receive).

4.1.1.2 Solaris

Unpack the archive into **/usr/local/sbin** or **/usr/local/bin**. This creates four files: **Dependencies.txt**, **ag_patchd**, **server.pl** and **Server.pm**. Rename **server.pl** to **ag_patchd_server.pl** to avoid any confusion. Edit the file to add as many daemon instances as you require to support the port forwards configured on the AppGate server. NB make sure that the path name to the executable daemon is correct – by default it is **/usr/local/bin**, so adjust it if you have installed the executable somewhere else.

Check **Dependencies.txt** to see that all required shared libraries are accessible on the library search path. You'll also need the Perl interpreter to run **server.pl**.

You need to arrange to run **ag_patchd_server.pl** automatically when your machine starts up. There are a number of strategies for doing this, including placing symbolic links to the script into **/etc/init/rc*.d**, running a periodic watchdog under cron and listing the script at the end of **/etc/rc.sysinit**. How you do it depends on the system management policy of your server. The script takes no parameters (i.e. it will ignore any that it does receive).

4.1.1.3 Other UNIX

Unpack the tarball archive into a developer directory, e.g. **~/Projects** or **/usr/src**. Change directory to the **ag_patchd** directory thus created. Then execute the following commands:

```
./configure
make
make install
```

Copy the three files **ag_patchd**, **server.pl** and **Server.pm** into **/usr/local/sbin** or **/usr/local/bin**, renaming **server.pl** to **ag_patchd_server.pl** to avoid any confusion. Edit the file to add as many daemon instances as you require to support the port forwards configured on the AppGate server. NB make sure that the path name to the executable daemon is correct – by default it is **/usr/local/bin**, so adjust it if you have installed the executable somewhere else.

You need to arrange to run **ag_patchd_server.pl** automatically when your machine starts up. There are a number of strategies for doing this, including placing symbolic links to the script into **/etc/init/rc*.d**, running a periodic watchdog under cron and listing the script at the end of **/etc/rc.sysinit**. How you do it depends on the system management policy of your server. The script takes no parameters (i.e. it will ignore any that it does receive).

4.2 Server Configuration

The **ag_patchd_server.pl** configuration on the AppGate Server is used to launch one **ag_patchd** instance for each port to be forwarded. If an instance terminates for any reason then the script will start another instance immediately.

The AppGate Console is used to define the set of port forwardings supported by **ag_patchd** as a service configuration (called PVPN or Personal VPN). It is then used to add and remove users and to provide them with access to the PVPN service. A relatively easily-to-manage port numbering scheme is suggested as follows. For each conventional protocol port N, assign

- N+20000 for the office client
- N+10000 for the mobile client

For example, to support POP3 (conventionally on port number 110) the mobile client would forward local port 110 to port 10110 on the AppGate Server, while the office client would forward local port 20110 to port 20110 on the AppGate Server. The script **ag_patchd_server.pl** on the AppGate server has to be configured to launch an **ag_patchd** instance with these port numbers. The Office Client on the user’s PC would be configured to connect to port 20110 on the AppGate Client, while forwarding data received on this connection to port 110 on the appropriate mail server.

The “translated„ port number is used on the office client to avoid clashes with real servers running on the user’s PC.

However, not all IP stacks support port numbers above 16383. In such cases, adjust the port number forwarded by the workstation AppGate client by e.g. subtracting 16384.

4.3 User provisioning and configuration

A process needs to be defined for users to join a trial programme and perform an honest evaluation of the results via a web form at the end of the trial, which should end after a fixed period of time. This is under development.

4.4 Client Software installation and configuration

4.4.1 Overview

Each user downloads and installs the two or three programs for the workstation separately. In future these might be supplied as a single combined installation.

When the software has been installed, the user calls up the Office Client and define servers for SMTP and POP3 or IMAP (giving access to mail, contacts, to-do lists and calendars), HTTP proxy and optionally other protocols. The configuration must reflect the set of port forwardings supported by the AppGate and **ag_patchd** configurations.

Each user downloads and installs the AppGate Client Lite (AGClite) appropriate to their mobile device. The mobile client needs to be configured to connect to the correct AppGate Server using the same user ID and password as the workstation AppGate Client. Applications such as mail clients on the device need to be configured to use localhost as their server (alternatively, AppGate can rewrite the /etc/hosts file to override the DNS entry for the server). Web browsers on the device need to be configured to use localhost and port 8080 as their proxy server. For further information on downloading and configuring the mobile client, please refer to AppGate documentation.

4.4.2 **ag_patchd** Office Client Installation

Download the archive for your machine architecture. Currently the client has been built for MacOS X, Linux (Intel 386 compatible CPUs) and Windows. A Java VM is not included (however, it is included in the AppGate Client, so you can use that). Although Java runs anywhere, this application is built using the SWT (standard widget toolkit), which relies on a small library of native code to interface Java to the native windowing system API.

4.4.2.1 Windows

Unzip the archive. It creates the folder **AgPatchd** at the top level of the C drive. The script **OfficeClient.bat** within this folder runs the client. You can place a shortcut to this on your desktop or the START menu, but this should be unnecessary, as the AppGate client should start it automatically if it is installed in the right place.

4.4.2.2 MacOS

Unpack the archive by dropping the file on StuffIt Expander. Choose the destination folder **/Applications/Internet**. Double-click the jar file to start it, or let the AppGate client start it automatically for you.

4.4.2.3 Linux

Unzip the archive into your home directory, or (if you have root privileges) into **/usr/local**. Create a script named **OfficeClient**, somewhere on your search path, with the following content:

```
exec .../AgPatchd/OfficeClient $@
```

(where "…„ stands for the path to the **AgPatchd** folder). Start the client by invoking this script. The AppGate client will start the client by calling your script.

4.4.3 Configuring the **ag_patchd** Office Client

Start the Office Client manually. By default, it displays parameters for a single service connection. You can add further services using the “Add service„ button. Any service can be deleted using the corresponding “Delete service„ button.

For each service, fill in a name that tells you what it does – e.g. HTTP, POP, SMTP or Exchange. Then fill in the fully qualified domain name or IP address of the corresponding server (e.g. **pop3.mydomain.com** or 10.1.20.44). In the case of HTTP, you need to specify the address of a proxy server. If you don’t know of any in your local network (check your browser preferences to see), install a local one on your workstation and specify “localhost„.

For each service, type in the port number on which the intranet server services requests – e.g. 25 for SMTP, 110 for POP3, 143 for IMAP4, 80 or 8080 for HTTP proxy (your browser preferences should tell you which).

Lastly, for each service type in the port number on which **ag_patchd** is forwarding packets from the mobile client. By convention, this is the server port number plus 20,000. In the case of HTTP, use the value 28080, regardless of the port number of the actual proxy server you are using.

The configuration of each service is saved to disk when you click the checkbox to activate it. Connect the AppGate Client to the AppGate server first – otherwise your system log will start to fill up with error messages (“connection refused„). In future, whenever you connect the AppGate client to the server and log in with your user ID, the Office Client will be started automatically and load the service configuration you last saved. Any services that were activated at the time you last quit the Office Client will automatically be activated.

The service configuration is stored in the editable XML file **.OfficeClient** in your home directory or folder.

4.5 In case of difficulty

In the first instance, please send e-mail to support@objectvalue.com. If there is a high volume of support requests, a knowledgebase will be set up under http://objectvalue.com/ and you will be requested to refer to this before you ask for more specialised support.