Blue Coat® Systems

Controlling Skype with the ProxySG Appliance

Blue Coat

CONTROL IS YOURS™

Americas:

Blue Coat Systems, Inc.
420 N. Mary Ave.
Sunnyvale, CA 94085

Rest of the World:

Blue Coat Systems International SARL
3a Route des Arsenaux
1700 Fribourg, Switzerland

Document History

Date	Version	Note
January 15, 2013	v0.1	Initial release

Contents

List of Figures

List of Tables

Introduction

Skype is both a proprietary Voice-over-Internet Protocol (VoIP) service and a software application that was originally created in 2003, and currently owned by Microsoft since 2011.

The Skype service allows users to communicate with peers by voice, video, and instant messaging over the Internet. Phone calls may be also placed to recipients on the traditional telephone networks. Calls to other users within the Skype service are free of charge, while calls to landline telephones and mobile phones are charged using a debit-based user account system. Skype also provides an Online Telephone Number service (known as SkypeIn), which allows Skype users to receive calls placed from traditional phones directly on their Skype client software application of choice.

Skype has become popular for its many additional features, including file transfer, and video conferencing, as well as for the Skype software application being made available for a plethora of operating systems, including those of mobile devices. Skype has also delivered specific service options for the Enterprise market.

These features have made Skype a valuable tool for Enterprises worldwide, while at the same time, making Skype a popular application for personal and recreational use. Many network administrators want to enable controls over Skype usage on corporate, government, and education networks, citing reasons such as inappropriate usage of resources, excessive bandwidth usage, and security concerns. As a result, implementing mechanisms for controlling the use of the Skype software application and subsequent Skype traffic are now needed.

How Skype Works

Unlike most VoIP services, Skype is a hybrid peer-to-peer and client–server system. Skype's original proposed name (Sky Peer-to-Peer) reflects this fact.

Since its introduction in 2003 and until very recently, the Skype service has been mainly based on the so-called, "supernodes," that were made up of regular users who had sufficient bandwidth, processing power, and other system requirements to qualify. These supernodes then transferred data with other supernodes in a peer-to-peer fashion. At any given time, there were typically just over 48,000 clients that operated this way.

However, at the time of this writing, Microsoft has drastically overhauled the network running its Skype VoIP service, replacing peer-to-peer client machines with thousands of Linux appliances that have been hardened against the most common types of attacks. In addition to hardening the appliances to hacks, the Microsoft-hosted supernodes are able to accommodate significantly more users. Supernodes under the old system typically handled about 800 end-users, whereas the newer systems can host about 4,100 users and have a theoretical limit of as many as 100,000 users.

Although Microsoft has not yet confirmed this data, slightly more than 10,000 supernodes that are all hosted by Microsoft could be powering Skype at the time of this writing. Therefore, it is currently not possible for regular users to be promoted to supernode status, which simplifies the implementation of selective Skype control in the ProxySG appliance, without the risk of having an internal

user unwittingly becoming a supernode. However, this has not changed the underlying nature of Skype's peer-to-peer (P2P) architecture, in which supernodes simply allow users to find one another. (Calls do not pass through supernodes.)

From the software client point of view, Skype is an extremely evasive and self-healing protocol. The Skype protocol is able to automatically react to changes in traffic restriction policies, to ensure that the Skype client software will be able to log in the user to the Skype service even in the most restrictive network environments. As long as the local network is connected to the Internet, the Skype client software and the Skype protocol itself have been designed to find a way to bypass traffic policy enforcement mechanisms in place within the Enterprise. This makes controlling Skype traffic a challenge, especially when restricting Skype usage to only a number of users is desired.

In a typical corporate network, a potential Skype user is connected to the Internet through an edge security layer consisting of at least a NAT/PAT device, a firewall (in the form of either a dedicated appliance or implemented as traffic inspection rules in routers), and a forward proxy. (See Figure 1–1.)

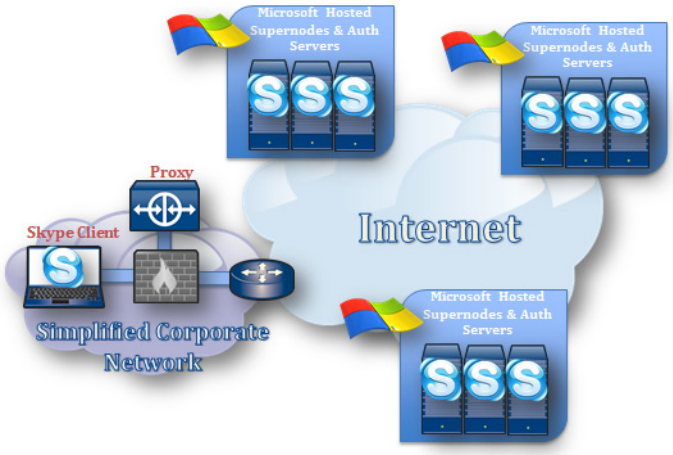

Figure 1–1 Simplified diagram on the current Skype overlay network

With the default Skype client software configuration, when a Skype user in the corporate network clicks the Skype **Sign me in** button, the Skype client initially attempts to directly connect to a list of predefined hosts and ports (Skype authentication servers and Skype supernodes). If this connection fails, (as it should in the case of an average secured corporate network), the Skype client software tries to access Skype hosts through well-known TCP ports, such as TCP/80 and TCP/443. If a forward proxy is not in place, these ports will be allowed as destination ports for egress traffic by the corporate edge security layer.

If this new connection attempt is unsuccessful, the Skype client software assumes that Internet access might be restricted to certain devices only, such as a proxy server, and automatically looks for an explicit proxy server configuration in the local computer in which the client software is running. In Microsoft Windows desktops, for instance, the Skype client checks the explicit proxy configuration within Internet Explorer.

If an explicit proxy configuration is discovered, Skype attempts to make its connections through the discovered explicit proxy, disguising the Skype protocol itself as HTTPS, by requesting to establish an SSL connection through HTTP Connect methods sent to the explicit proxy.

If the existing proxy is not able to properly handle SSL traffic at the application layer (for example, if the proxy itself is not an SSL proxy or Connect proxy) or if a Blue Coat ProxySG appliance (which is an SSL proxy/Connect proxy) is not correctly configured, Skype will find its way out of the corporate network, and will successfully log in the user to the Skype overlay network.

This Technical Brief describe how to correctly configure a ProxySG appliance deployed in explicit proxy mode in order to not only block Skype traffic completely, but also selectively allow Skype traffic with in the corporate network for specific users.

Requirements

Figure 1–2 illustrates the simplified corporate network that will be used in the following sections to demonstrate the implementation of granular Skype controls in a ProxySG appliance.

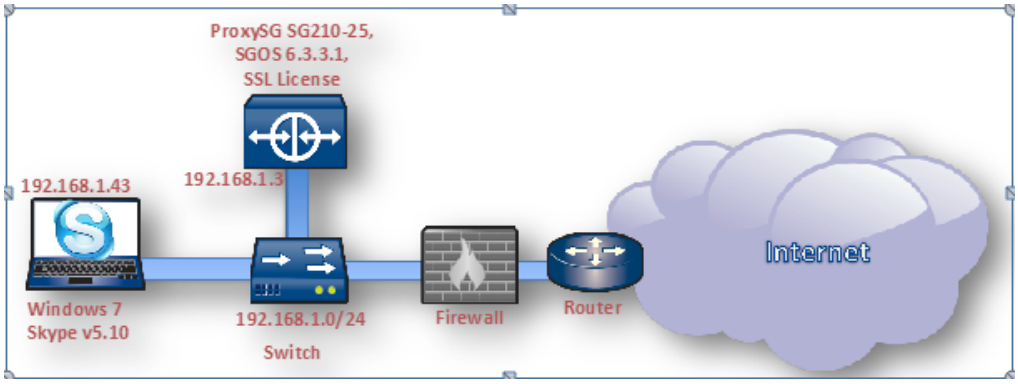

Figure 1–2 Simplified corporate network

As illustrated in Figure 1–2, to demonstrate the contents of this Technical Brief, we will be using:

- A Windows 7 laptop, running Skype v5.10.0.116 with the factory default configuration. (See Figure 1–3.)
- The Internet Explorer browser in this laptop has been configured to use the explicit proxy in IP address 192.168.1.3, port TCP/8080.

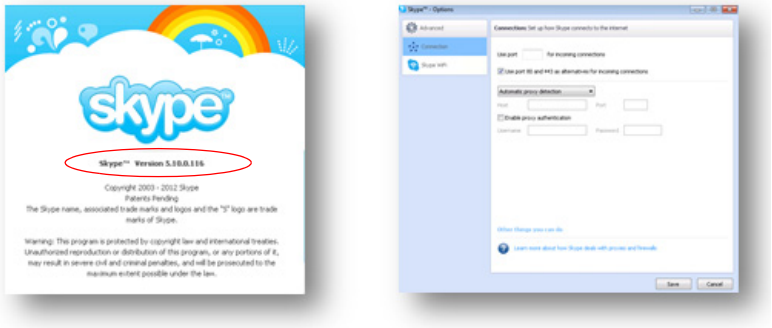

Figure 1–3 Skype Version 5.10.0.116 with default configuration

- A firewall, which has been configured so that the ProxySG appliance (192.168.1.3) is the only host allowed to directly connecting to the Internet (except for DNS traffic, which is allowed from all host in the trusted/internal network).
- A Layer-2 switch, interconnecting the client laptop, the ProxySG appliance, and the Firewall.
- An edge router for Internet connection, with PAT enabled, so that all corporate Internet traffic will be multiplexed at the IP layer, over a single public IP address.
- A ProxySG appliance (SG210-25-PR) with the Full Proxy and an active SSL license.

For the purpose of this document, the ProxySG appliance is running SGOS 6.3.3.1. All configurations detailed in this document can be directly applied to any ProxySG appliance running SGOS 6.2.x and 6.3.x, except for minimal changes in the graphical user interface.

An active SSL license is mandatory in order to control Skype traffic, as well as traffic generated by any other application trying to bypass an HTTP proxy using HTTP Connect methods to mimic HTTPS traffic.

For the remainder of this document, it is assumed that the corporate Internet firewall will be configured so that the ProxySG appliance is the only host allowed to directly connecting to the Internet. For example, only those packets with a source IP address—that of the ProxySG appliance—will be granted Internet access. Otherwise, the Skype software client will be able to directly connect to the Internet (as described earlier in this document) and Skype controls defined in the ProxySG appliance would simply not apply.

To cope with this requirement, the Firewall in Figure 1–2 has been configured so that only packets sourced at 192.168.1.3 are allowed from the trusted network to the untrusted network, except for DNS traffic, which is allowed from any host (including the host running Skype software: 192.168.1.43).

If the ProxySG appliance does not have an active SSL license, or if the explicit HTTP proxy service (under **Configuration > Services > Proxy Services**) has the **Detect Protocol** option disabled, (as shown in Figure 1–4), the Skype client software is able to bypass the ProxySG appliance, as is demonstrated in Table 1–2 on page 6.

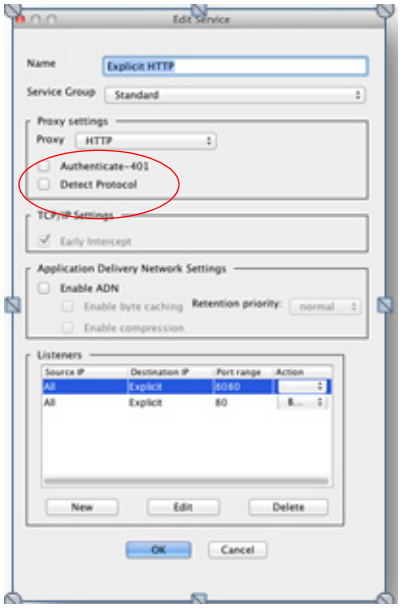

Figure 1–4 Detect Protocol option disabled within the explicit HTTP proxy

As described earlier in this document, the Skype client software first attempts to make direct connections to Skype servers. In the test environment illustrated in Figure 1–2, these attempts are blocked by the firewall, as is shown in the following firewall logs:

```
2012-08-01 14:42:42   192.168.1.43:51489  65.54.52.42:443       HTTPS            Traffic Denied
2012-08-01 14:42:36   192.168.1.43:51489  65.54.52.42:443       HTTPS            Traffic Denied
2012-08-01 14:42:36   192.168.1.43:36718  157.55.56.149:33033   UDP PORT 33033   Traffic Denied
2012-08-01 14:42:34   192.168.1.43:36718  157.56.52.25:33033    UDP PORT 33033   Traffic Denied
2012-08-01 14:42:33   192.168.1.43:51489  65.54.52.42:443       HTTPS            Traffic Denied
2012-08-01 14:42:33   192.168.1.43:36718  111.221.74.23:33033   UDP PORT 33033   Traffic Denied
2012-08-01 14:42:32   192.168.1.43:36718  65.55.223.20:33033    UDP PORT 33033   Traffic Denied
2012-08-01 14:42:31   192.168.1.43:36718  157.56.52.18:33033    UDP PORT 33033   Traffic Denied
2012-08-01 14:42:30   192.168.1.43:36718  157.55.56.158:33033   UDP PORT 33033   Traffic Denied
2012-08-01 14:42:13   192.168.1.43:51488  64.4.23.162:80        HTTP             Traffic Denied
2012-08-01 14:42:12   192.168.1.43:51487  64.4.23.162:443       HTTPS            Traffic Denied
2012-08-01 14:42:10   192.168.1.43:51486  64.4.23.162:33033     TCP PORT 33033   Traffic Denied
2012-08-01 14:42:07   192.168.1.43:51488  64.4.23.162:80        HTTP             Traffic Denied
2012-08-01 14:42:06   192.168.1.43:51487  64.4.23.162:443       HTTPS            Traffic Denied
2012-08-01 14:42:04   192.168.1.43:51488  64.4.23.162:80        HTTP             Traffic Denied
2012-08-01 14:42:04   192.168.1.43:51486  64.4.23.162:33033     TCP PORT 33033   Traffic Denied
2012-08-01 14:42:03   192.168.1.43:51487  64.4.23.162:443       HTTPS            Traffic Denied
2012-08-01 14:42:01   192.168.1.43:51486  64.4.23.162:33033     TCP PORT 33033   Traffic Denied
2012-08-01 14:42:00   192.168.1.43:36718  157.55.56.143:33033   UDP PORT 33033   Traffic Denied
2012-08-01 14:41:59   192.168.1.43:36718  157.55.130.154:33033  UDP PORT 33033   Traffic Denied
```

Table 1–1 Firewall logs

At this point, the Skype client has concluded that direct Internet connections are being blocked and initiates proxy detection. In this case, the Skype clients reads the explicit proxy configuration from Internet Explorer and makes a connection attempt through the explicit HTTP proxy using the HTTP Connect method. This simulates an HTTPS browsing session, which is allowed through the proxy because the current ProxySG configuration either has no active SSL license or "Detect Protocol" is disabled in the explicit HTTP proxy. Traffic is processed as a TCP tunnel (session bridging at the Transport Layer) without any further checks. This is demonstrated in Table 1–2, which lists the ProxySG appliance access log entries from the main log (s-action: `TCP_Tunneled` with s-method: `Connect`).

2012-08-01 14:41:17	4623	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	157.55.56.152	443	-	192.168.1.3
2012-08-01 14:41:30	11275	192.168.1.43	OBSERVED	None	200	TCP_TUNNELED	CONNECT	tcp	195.95.154.39	443	-	192.168.1.3
2012-08-01 14:41:32	601	192.168.1.43	OBSERVED	Internet Telephony	200	TCP_TUNNELED	CONNECT	tcp	212.161.8.36	443	-	192.168.1.3
2012-08-01 14:41:33	10711	192.168.1.43	OBSERVED	Internet Telephony;Chat/Instant Messaging	200	TCP_CLIENT_REFRESH	GET	http	ui.skype.com	80	Skype® 5.10	192.168.1.3
2012-08-01 14:41:33	1343	192.168.1.43	OBSERVED	Internet Telephony;Chat/Instant Messaging	200	TCP_TUNNELED	CONNECT	tcp	api.skype.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-01 14:41:33	2477	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	ajax.googleapis.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-01 14:41:33	3739	192.168.1.43	OBSERVED	Social Networking	200	TCP_TUNNELED	CONNECT	tcp	connect.facebook.net	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-01 14:41:33	525	192.168.1.43	OBSERVED	Internet Telephony	200	TCP_TUNNELED	CONNECT	tcp	212.161.8.36	443	-	192.168.1.3
2012-08-01 14:41:35	21762	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	65.55.223.25	443	-	192.168.1.3
2012-08-01 14:41:36	21096	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	65.55.223.31	443	-	192.168.1.3
2012-08-01 14:41:37	21153	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	111.221.77.140	443	-	192.168.1.3
2012-08-01 14:41:39	21857	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	157.55.56.161	443	-	192.168.1.3
2012-08-01 14:41:40	21457	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	111.221.77.154	443	-	192.168.1.3

Table 1–2 ProxySG appliance main access log (displaying only relevant fields)

Note: The table uses a tabular format for the sake of clarity.

Configuration

To configure the ProxySG appliance to block Skype traffic, complete the following procedure.

1. Enable **Detect Protocol** in the explicit HTTP proxy service. (See Figure 1–5.)
   a. From the ProxySG Management Console, select **Configuration > Services > Proxy Services**.
   b. Under **Proxy settings**, select the **Detect Protocol** check box.
2. Click **Edit** to edit the explicit HTTP proxy service.
   a. Make changes to the service as needed.
   b. Click **OK**.
3. Click **OK**.

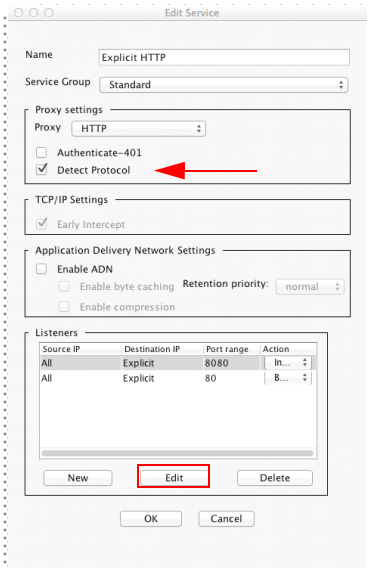

Figure 1–5 Edit Service dialog - enable Detect Protocol

4. Verify that the ProxySG appliance has an active SSL License.

   a. From the ProxySG Management Console, select **Maintenance > Licensing > View.**

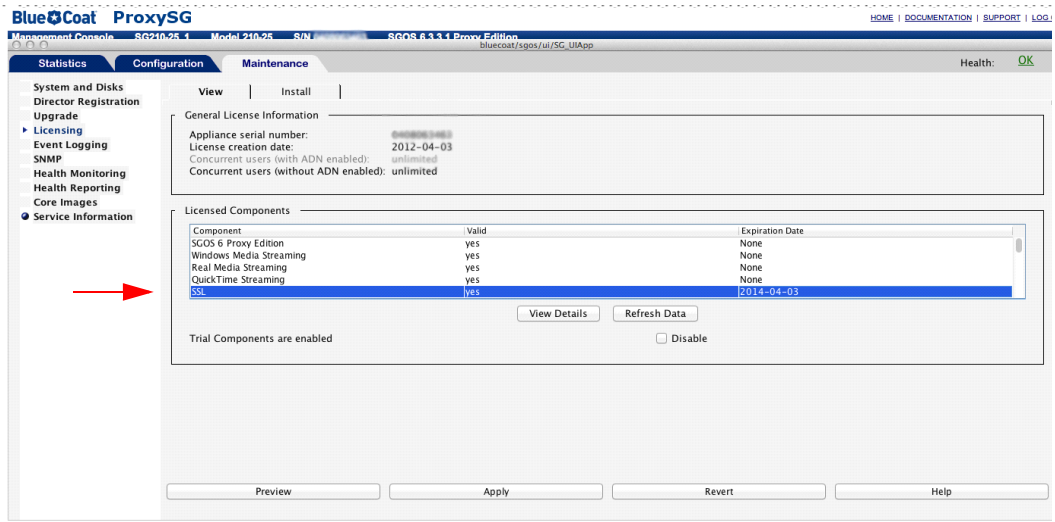

Figure 1–6 ProxySG appliance with an active SSL License

Based on this configuration, when the ProxySG appliance receives an HTTP request containing a Connect method, the ProxySG appliance will internally hand off the embedded traffic to the SSL worker process. This is demonstrated in Table 1–3 on page 8.

2012-08-01 15:56:43	363	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	65.55.223.21	443	192.168.1.3
2012-08-01 15:56:44	171	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	65.55.223.14	443	192.168.1.3
2012-08-01 15:56:45	118	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	213.199.179.145	443	192.168.1.3
2012-08-01 15:56:47	520	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	65.55.223.25	443	192.168.1.3
2012-08-01 15:56:48	111	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	111.221.77.143	443	192.168.1.3
2012-08-01 15:56:50	110	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	111.221.74.32	443	192.168.1.3
2012-08-01 15:56:51	111	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.56.52.22	443	192.168.1.3
2012-08-01 15:56:52	601	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.55.235.159	443	192.168.1.3
2012-08-01 15:58:32	234	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	111.221.77.141	443	192.168.1.3
2012-08-01 15:58:33	103	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.56.52.25	443	192.168.1.3
2012-08-01 15:58:34	152	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	111.221.77.160	443	192.168.1.3
2012-08-01 15:58:35	159	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.55.56.141	443	192.168.1.3
2012-08-01 15:58:36	106	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	213.199.179.146	443	192.168.1.3
2012-08-01 15:58:38	121	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.55.130.166	443	192.168.1.3
2012-08-01 15:58:39	147	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.55.130.148	443	192.168.1.3
2012-08-01 15:58:40	110	192.168.1.43	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	157.55.235.165	443	192.168.1.3

Table 1–3 ProxySG appliance main access log (simplified) when configured to block Skype traffic

Note: When the explicit HTTP proxy receives an HTTP Connect tunnel connection and protocol detection is enabled, if the ProxySG appliance has an internal worker process for the protocol detected (as it is the case for SSL), the ProxySG appliance will hand the connection off to the appropriate workers to perform additional inspection and processing. If the handoff is successful, s-action: `TCP_ACCELERATED` is logged in the log record of the original tunnel transaction.

The SSL worker will then realize that the alleged SSL session initiated by the Skype client is not a valid SSL session, since the destination server is not presenting a valid SSL certificate. (Actually, the servers targeted by the Skype client software will not present any certificate at all.) This will instruct the ProxySG appliance to discard the connection attempt, and ultimately make the Skype client software report the inability to log in the user to the Skype service.

The SSL debug log (shown below) provides evidence of this behavior, which shows an excerpt from the SSL Debug output. The output demonstrates the Skype client software attempt to connect to the Skype service through SSL, which is being blocked due to a “Missing SSL server certificate” error.

Note: The SSL Debug messages are presented in reverse order (the most recent debug message first), and should, therefore, be read from the bottom up.

```
8214.822 SSLW FE0584AC (B6C00201): Remove byte-record-bio from read bio
8214.822 SSLW FE0584AC (B6C00201): Remove byte-record-bio from read bio
8214.820 SSLW FE0584AC (B6C00201): Tunnel decision for ssl://
157.55.56.148:443/
8214.819 SSLW FE0584AC (B6C00201): Tunnel "Missing SSL server certificate"
error for ssl://157.55.56.148:443/
```

```
8194.770 SSLW FE0584AC (B6C00201): upstream hit = 0, server_hit = 0
8194.520 SSLW FE0584AC (B6C00201): Executing ssl connect to: ssl://
157.55.56.148:443/
8194.520 SSLW FE0584AC (B6C00201): Enter ssl_set_byte_record_bio
8194.520 SSLW FE0584AC (B6C00201): Setup upstr ssl, ssl=331B6454
8194.520 SSLW FE0584AC (B6C00201): cloned SSL version = 301
8194.520 SSLW FE0584AC (B6C00201): Cloning upstr session
8194.520 SSLW FE0584AC (B6C00201): dwnstr_clnt_hello_type: E7
8194.327 SSLW FE0584AC (B6C00201): Executing ssl accept
8194.327 SSLW FE0584AC (B6C00201): Setup dwnstr ssl, ssl=331B68A4, byte
record bio=F5DBB454
8194.327 SSLW FE0584AC (B6C00201): Enter ssl_set_byte_record_bio
8194.327 SSLW FE0584AC (B6C00201): Retrieving ctx from server_map
8194.327 SSLW FE0584AC (B6C00201): SSL Intercept URL: "ssl://
157.55.56.148:443/"
[…] output omitted […]
8216.822 SSLW FE0604AC (4A4001ED): Remove byte-record-bio from read bio
8216.822 SSLW FE0604AC (4A4001ED): Remove byte-record-bio from read bio
8216.819 SSLW FE0604AC (4A4001ED): Tunnel decision for ssl://
157.55.235.166:443/
8216.819 SSLW FE0604AC (4A4001ED): Tunnel "Missing SSL server certificate"
error for ssl://157.55.235.166:443/
8195.878 SSLW FE0604AC (4A4001ED): upstream hit = 0, server_hit = 0
8195.716 SSLW FE0604AC (4A4001ED): Executing ssl connect to: ssl://
157.55.235.166:443/
8195.716 SSLW FE0604AC (4A4001ED): Enter ssl_set_byte_record_bio
8195.716 SSLW FE0604AC (4A4001ED): Setup upstr ssl, ssl=331B6A14
8195.716 SSLW FE0604AC (4A4001ED): cloned SSL version = 301
8195.716 SSLW FE0604AC (4A4001ED): Cloning upstr session
8195.716 SSLW FE0604AC (4A4001ED): dwnstr_clnt_hello_type: E7
8195.603 SSLW FE0604AC (4A4001ED): Executing ssl accept
8195.603 SSLW FE0604AC (4A4001ED): Setup dwnstr ssl, ssl=331B65C4, byte
record bio=F5DBB274
8195.603 SSLW FE0604AC (4A4001ED): Enter ssl_set_byte_record_bio
8195.602 SSLW FE0604AC (4A4001ED): Retrieving ctx from server_map
8195.602 SSLW FE0604AC (4A4001ED): SSL Intercept URL: "ssl://
157.55.235.166:443/"
```

Blue Coat's Knowledge Base FAQ448 details the process for enabling and accessing the SSL Debug Log.

Note

In some SGOS releases, SSL handshake failures/SSL certificate-related errors are logged into the event log. For example:

```
"SSL client handshake failure"
"SSL server handshake failure"
"Missing SSL server certificate"
```

Although these SSL-related error messages are based on genuine SSL traffic, too many of them can prevent you from seeing other important events in the event log. For example, the log can quickly fill with SSL handshake and certificate-related errors whenever a client or a server drops the connection in the middle of the SSL handshake. Therefore, starting in SGOS 6.2.8.1 and 6.3.3.x, these error messages are logged to the SSL debug log instead of the event log.

As a result of this configuration, Skype software client will not be able to connect the user to the Skype service. (See Figure 1–7.)

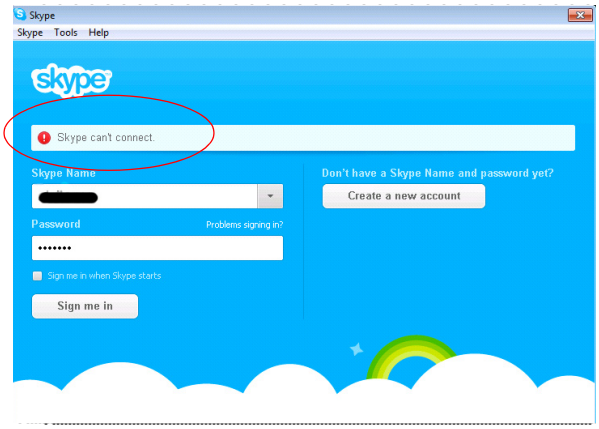

Figure 1–7 Skype can't connect message

Setting User Policies to Selectively Allow Skype

The internal process implementing protocol detection on top of the explicit HTTP proxy can be made conditional in the ProxySG appliance. Using this method, it's possible to block Skype for some users, while selectively allowing it for others.

Starting with SGOS 6.2, the policy conditions to allow this level of granular control are defined using the Visual Policy Manager (VPM) in the ProxySG appliance Admin Console.

From the VPM, you can use various parameters to allow or disallow Skype traffic (for example, user, IP address, time of day). A common scenario might be to allow Skype for a small subset of users and disallow it for everyone else.

From this point on, we will assume that the ProxySG appliance has been previously configured for user authentication by creating the necessary Authentication Realms and by configuring the respective Web Access Layer.

Configuration Steps

To configure the ProxySG appliance to *selectively* allow Skype traffic, complete the following procedure.

Procedure:

1. Launch the VPM from the ProxySG Management Console. To do this, select **Configuration > Policy > Visual Policy Manager.**
2. Create a new Web Access Layer (or select an existing one in your VPM), and add a new rule to selectively disable protocol detection for Connect methods over HTTP. (See Figure 1–8 on page 12.)
   a. Select the **Web Access Layer** tab.
   b. Right-click the **Source** field of the newly created rule and select **Set**.
   c. In the **Set Source Object** dialog, click **New** and select **Combined Source object**.
   d. Name the object. The example is using AllowedSkypeSources.
   e. Select existing user objects representing the users for whom to allow Skype access (or create new user objects as needed) and add them to the top right **At least one of these objects** window by clicking **Add >>**.
   f. Within the same dialog, create a new **Request Header Object** by clicking **New**. As shown in Figure 1–8, this request header object will match on HTTP requests not containing the User-Agent request header field (requested by a regular expression matching the empty string: "^$").
   g. Add the newly created request header object to the bottom right **At least one of these objects** window by clicking **Add >>**. This restricts the entire condition to only HTTP requests not containing a User-Agent field.

      This configuration allows the control of Skype, while having SSL interception enabled for actual HTTPS traffic and for those users allowed to use Skype. (Otherwise, HTTPS traffic is never intercepted for those users allowed to connect to the Skype service.)

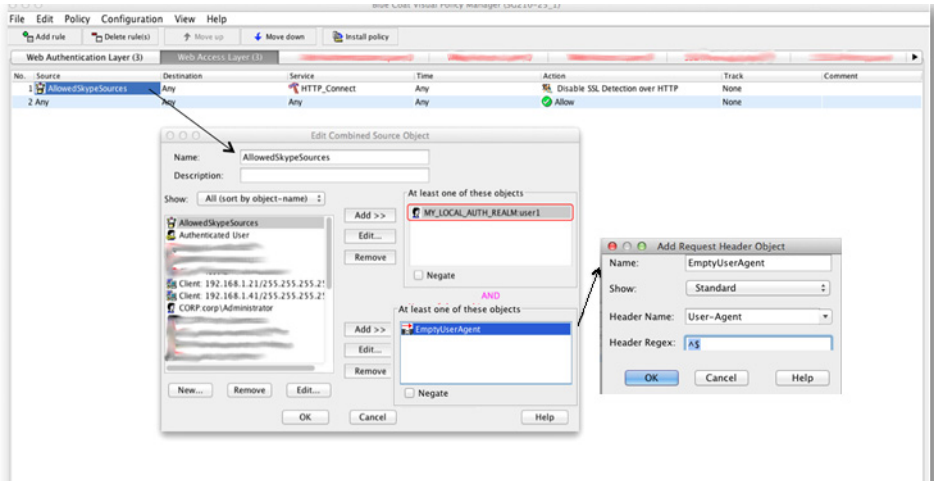

Figure 1–8 Selectively allowing Skype - Source conditions: a combined object consisting of a list of allowed users and the negated list of User Agents

3. Set service to apply only to HTTP Connect. (See Figure 1–9.)
   a. Right-click the **Service** field, then select **Set**.
   b. From the **Edit** dialog, define the service condition to apply only to connect methods over HTTP.
   c. Click **OK**.

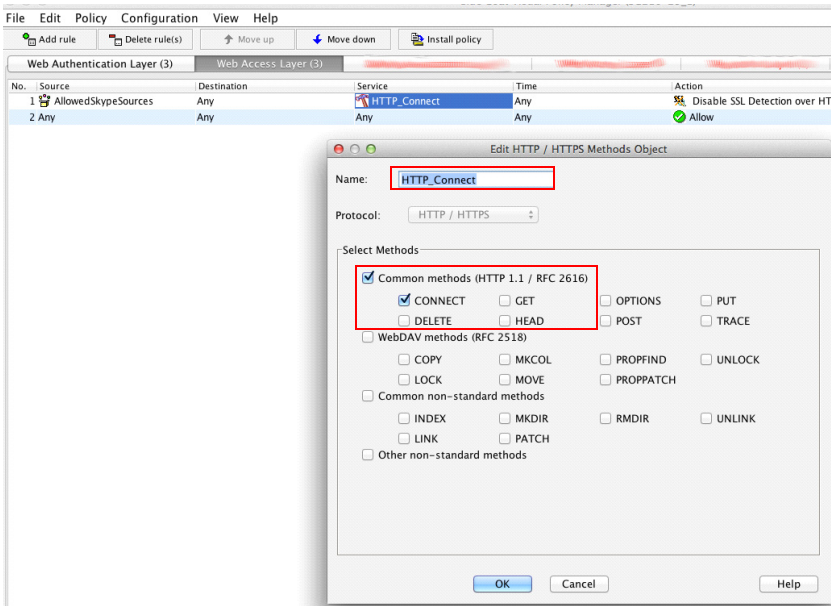

Figure 1–9 Selectively allowing Skype - Service conditions

4. Set action parameters. (See Figure 1–10.)
   a. Right-click the **Action** field and select **Set > Disable SSL Detection**.
   b. From the **Edit Disable SSL Detection Object** dialog, disable SSL detection for traffic tunneled over HTTP Connect.
   c. Click **OK**.

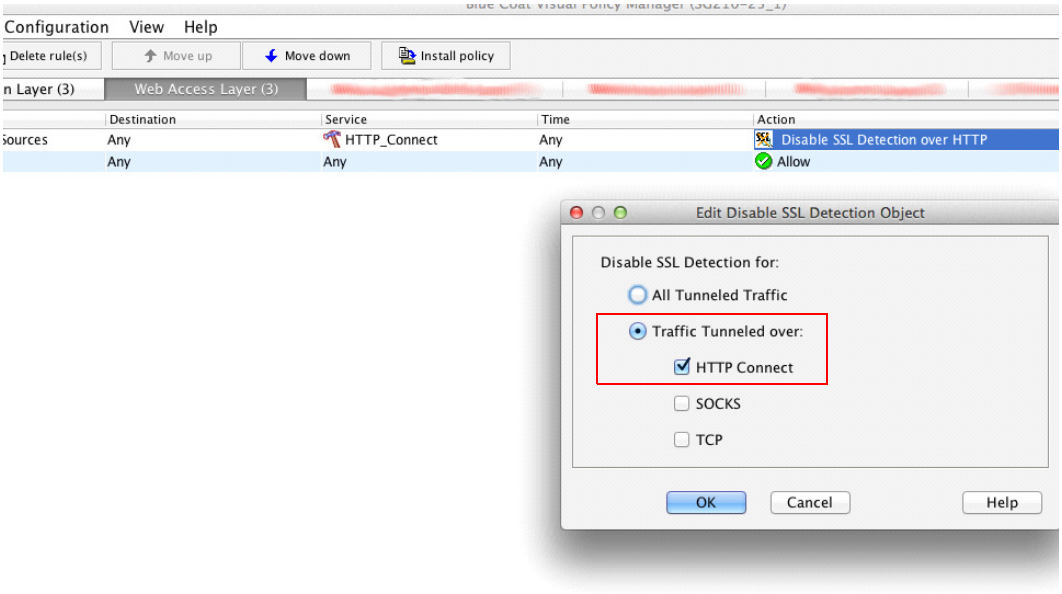

Figure 1–10 Selectively allowing Skype - Action

With this configuration, only the user "user1" is allowed to connect to the Skype service through the ProxySG appliance. (See Figure 1–11).

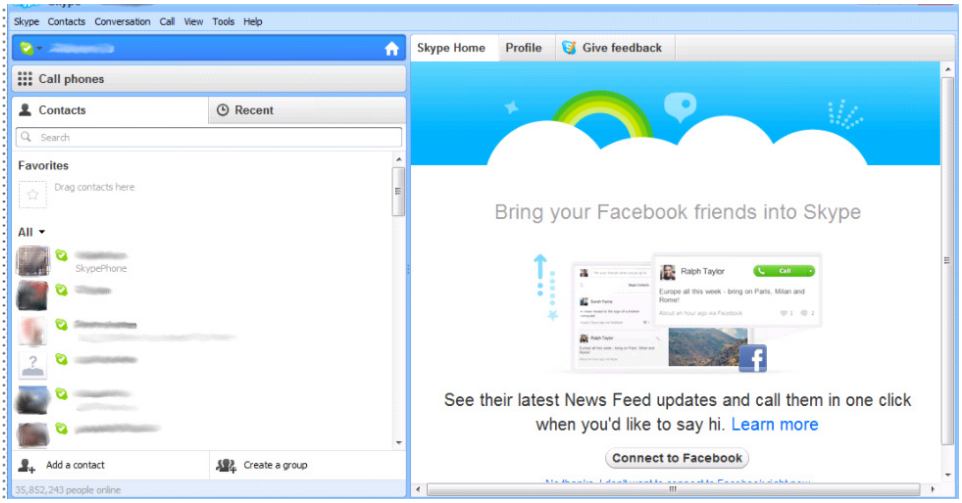

Figure 1–11 Connection to Skype service through ProxySG appliance allowed for "user1"

Depending on the authentication mechanisms used in the ProxySG appliance, you might be required to configure Skype with proxy authentication credentials. (See Figure 1–12.)

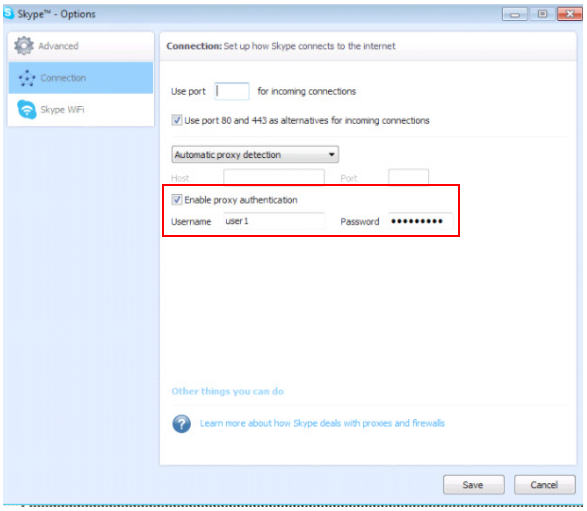

Figure 1–12 Skype client - proxy authentication credentials

Table 1–4 on page 15 shows the relevant information from the main access log in the ProxySG appliance, demonstrating how user1 successfully connects to the Skype service.

2012-08-02 08:22:59	192.168.1.43	user1	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	111.221.77.149	443	-	192.168.1.3
2012-08-02 08:23:05	192.168.1.43	user1	OBSERVED	Internet Telephony;Chat/Instant Messaging	200	TCP_ACCELERATED	CONNECT	tcp	apps.skype.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:06	192.168.1.43	user1	OBSERVED	Internet Telephony;Chat/Instant Messaging	200	TCP_CLIENT_REFRESH	GET	http	ui.skype.com	80	Skype6 5.10	192.168.1.3
2012-08-02 08:23:06	192.168.1.43	user1	OBSERVED	Social Networking	200	TCP_ACCELERATED	CONNECT	tcp	connect.facebook.net	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:07	192.168.1.43	user1	OBSERVED	Computers/Internet	200	TCP_ACCELERATED	CONNECT	tcp	ajax.googleapis.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:08	192.168.1.43	user1	OBSERVED	Internet Telephony;Chat/Instant Messaging	200	TCP_ACCELERATED	CONNECT	tcp	api.skype.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:10	192.168.1.43	user1	OBSERVED	Social Networking	200	TCP_ACCELERATED	CONNECT	tcp	s-static.ak.facebook.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:10	192.168.1.43	user1	OBSERVED	Social Networking	200	TCP_ACCELERATED	CONNECT	tcp	www.facebook.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:10	192.168.1.43	user1	OBSERVED	Non-viewable	200	TCP_ACCELERATED	CONNECT	tcp	c.msn.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:10	192.168.1.43	user1	OBSERVED	Web Advertisements	200	TCP_ACCELERATED	CONNECT	tcp	aidps.atdmt.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:11	192.168.1.43	user1	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	212.187.172.78	443	-	192.168.1.3
2012-08-02 08:23:12	192.168.1.43	user1	OBSERVED	Web Advertisements	200	TCP_ACCELERATED	CONNECT	tcp	aidps.atdmt.com	443	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C)	192.168.1.3
2012-08-02 08:23:12	192.168.1.43	user1	OBSERVED	Internet Telephony	200	TCP_TUNNELED	CONNECT	tcp	204.9.163.200	443	-	192.168.1.3
2012-08-02 08:23:14	192.168.1.43	user1	OBSERVED	Internet Telephony	200	TCP_TUNNELED	CONNECT	tcp	204.9.163.200	443	-	192.168.1.3
2012-08-02 08:23:19	192.168.1.43	user1	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	157.55.56.159	443	-	192.168.1.3
2012-08-02 08:23:20	192.168.1.43	user1	OBSERVED	Email	200	TCP_TUNNELED	CONNECT	tcp	64.4.23.166	443	-	192.168.1.3
2012-08-02 08:23:22	192.168.1.43	user1	OBSERVED	Computers/Internet	200	TCP_TUNNELED	CONNECT	tcp	157.55.235.159	443	-	192.168.1.3

Table 1–4 ProxySG appliance main access log (simplified) when configured to allow Skype traffic for “user1”

Based on the configuration provided in this document, Table 1–4 shows the ProxySG appliance selectively disabling SSL protocol detection over HTTP Connect methods only when the request doesn’t have a User-Agent field (s-action: `TCP_TUNNELED`). The ProxySG appliance is still performing protocol detection for the rest of the traffic, correctly handing off the HTTP Connect

method to the SSL proxy worker process for all remaining connect methods received (SSL protocol is detected), even those requests from the same user (s-action: `TCP_ACCELERATED`). This makes selective Skype control fully compatible with full HTTPS interception in the ProxySG appliance.

Conclusion

The Skype service, featuring one of the most evasive communication protocols to date, is quickly finding its place within the Enterprise user base due to its many features. In addition to Internet voice calls and instant messaging, Skype provides the means to support file transfers, desktop sharing, and group video conferencing. While Skype delivers specific service options for today's Enterprise markets, the Skype software client application has been made available for a plethora of operating systems, including those for mobile devices, such as iOS and Android.

Due to its many features, Skype has become a valuable tool for Enterprises worldwide, while at the same time, being a popular application for personal and recreational use, making it necessary to implement mechanisms for controlling the use of the Skype service in the Enterprise.

As the cornerstone of Blue Coat's Secure Web Gateway infrastructure, the ProxySG appliance not only provides rich policy controls for HTTP and HTTPS traffic, but also provides the required application-layer intelligence to control Skype. This, coupled with the well-known integration of ProxySG appliance and its authentication mechanisms in place on the Enterprise, provides ProxySG appliance administrators the ability to selectively allow the Skype protocol on a per-user basis.

About Technical Briefs

Technical briefs are designed to illustrate the features and capabilities of Blue Coat products. By describing generic solutions, technical briefs provide a foundation that Blue Coat customers can use to understand how Blue Coat products can be used to solve specific problems.

These technical briefs are not intended to solve customer-specific requests; if you need a customized solution to address a specific concern, contact Blue Coat Professional Services at Professional.Services@bluecoat.com.