



Document Library

Die InterLake GmbH bietet Ihren Kunden in der Document Library Zugriff auf interessante Informationen. Wir stellen Ihnen Inhalte unserer Partner und aus frei zugänglichen Online Quellen zur Verfügung, die mit unserer Tätigkeit und der IT- und Medienbranche zu tun haben. Bitte beachten Sie, dass die Inhalte dem Copyright der jeweiligen Herausgeber unterliegen und diese Inhalte nicht ohne Referenz auf das jeweilige Copyright weitergegeben werden dürfen.

Weitere Informationen zu InterLake und weitere Dokumente finden Sie unter

www.interlake.net

Die InterLake Document Library steht den Nutzern der InterLake.Network zur Verfügung:



InterLake engagiert sich ehrenamtlich beim Münchener IT- und Medienverband FIWM, dem unser Geschäftsführer Sven Slazenger als Vorstandsmitglied angehört, sowie in der Macromedia ColdFusion User Group Central Europe, die unter unserer Leitung seit 1998 ein deutschsprachiges Forum für über 700 Macromedia Entwickler in Deutschland, Österreich und der Schweiz bietet.

Weitere Informationen zu diesen beiden Initiativen erhalten Sie bei Sven Slazenger (slazenger@interlake.net)

InterLake offers its customers and business partners access to an extensive document library. We are offering information from our partners and have also compiled freely available papers from the internet that we consider of value to you. Please be advised that the copyright belongs to the issuer of the information and that you may not distribute this content without referring to these copyrights.

Additional information about InterLake and more documents can be accessed free of charge at

www.interlake.net

The InterLake Document Library is available through the websites of the InterLake.Network:

InterLake is also an active member of the non-profit Munich IT- and Media Association FIWM. With InterLake CEO Sven Slazenger we provide one of the board members of the FIWM. We are also the founders (1998) of the Macromedia ColdFusion User Group Central Europe, the German language forum for Macromedia Developers from Austria, Germany and Switzerland and one of the largest Macromedia Communities worldwide.

For further information please contact Sven Slazenger (slazenger@interlake.net)



Enabling Flash Communications with Tunneling

When the first version of the Flash Communication Server MX was released some developers discovered that they could not show corporate clients communication applications they had developed because their clients were behind firewalls and proxy servers. In some cases developers found they could work around the connection problem by making small changes in their application and/or reconfiguring the communication server. But, in some cases nothing worked. Macromedia's response was to add HTTP tunneling as a new feature in version 1.5. Tunneling is designed to solve many firewall and proxy server connection problems and introduces a surprising new feature that offers a new level of security to communication applications. This article describes tunneling, why it is necessary, and how to work with it. It is also designed to provide enough background information on firewalls and proxy servers so that you can make informed choices when writing communication applications and configuring the server. However, there are a lot of options for configuring and securing the Flash Communication Server MX, this article cannot possibly cover them all. Macromedia has provided a number of useful resources you should also consult:

- The full documentation for the server is available at <http://www.macromedia.com/support/flashcom/documentation.html>. The *Managing Flash Communication Server document* is something you should make sure you consult - especially the section named *Configuring Flash Communication Server*.
- David Simmons provides a security overview at http://www.macromedia.com/devnet/mx/flashcom/articles/security_overview.html
- Mike Chambers' white paper is also good background reading: <http://www.macromedia.com/devnet/mx/flash/whitepapers/security.pdf>
- Macromedia's technote on HTTP tunneling: http://www.macromedia.com/support/flashcom/ts/documents/http_tunneling.html
- Communication server 1.5 release notes: http://www.macromedia.com/support/flashcom/releasenotes/mx/rn_mx_1.html
- Finally, Phillip Kerman's article on the new features in version 1.5 of the Flash Communication Server MX introduces http tunneling: <http://www.macromedia.com/devnet/mx/flashcom/articles/exploring.html>.

Attacks and Firewalls

Firewalls are an essential tool in protecting networks from attacks. A firewall can pay for itself several times over by preventing a large number of attacks that otherwise can disable or corrupt production systems and waste many hours of IT staff's time in repairing and recovering compromised systems. The "Slammer" worm and the problems it caused in late January of 2003 is a good example of why organizations use firewalls. The worm exploits a vulnerability in certain versions of Microsoft's Microsoft SQL Server 2000 and Microsoft Desktop Engine 2000. The worm spreads by trying to access the resolution service of

the software by sending udp packets to port 1434 of randomly chosen IP addresses. Networks that blocked udp traffic on port 1434 were effectively immune from the worm. Networks that did not block the worm were infected if the worm found any system with unpatched versions of those two Microsoft products. Unfortunately, if network traffic coming in was not blocked by a firewall, infected systems were also unlikely to be blocked from spreading the worm to systems on other people's networks. The slammer worm only exploits one vulnerability in two products from one vendor. But there are many other vulnerabilities in many different operating systems and applications and new vulnerabilities are being discovered all the time. You only have to read the advisories at <http://www.cert.org/> to see the scale of the problem. Applying security patches to software with vulnerabilities is an important part of securing a network but it is only one part. A firewall is just as essential. The slammer worm also illustrates why IT security groups apply very restrictive rules when they setup firewalls. Some organizations with well protected and patched core production systems but weak firewall rules discovered that a few unpatched workstations or test systems completely disabled their entire network when the worm found them. Their core production systems were not directly affected by the worm but their core systems were still useless because all the network traffic the worm generated meant no one could connect to them..

A well-configured firewall will normally be configured with a set of rules that starts by denying all network traffic into and out of an organization's network. Then rules are applied that open access under a few carefully selected circumstances. The rules applied by any organization will be different depending on the needs of each organization. The rules will involve a number of settings:

- what IP addresses inside the firewall are reachable from the outside? If a computer on the Internet requests a connection to a machine on the local network will the request be passed through to the machine by the firewall or simply blocked?
- if an IP address inside the firewall is reachable, what type of traffic is permitted? TCP and/or UDP? TCP, or Transport Control Protocol, is the basic protocol on which many other common protocols such as HTTP are based. It is a reliable and bidirectional protocol. UDP, or User Datagram Protocol, is an unreliable protocol (there is no built-in guarantee all packets will arrive) that can be used to send information to one or many addresses. UDP is used by many streaming media servers. The Macromedia Flash Communication Server MX uses TCP.
- if an IP address inside the firewall is reachable using either UDP or TCP from the outside, what port or ports can be reached on that machine? Port numbers are the addresses of applications running on the machine. For example connecting to port 80 normally means a request will be made of a Web server if one is available. A request to port 443 will normally be to a system that provides an SSL service that encrypts data passed back and forth between the server and a remote client. Another common application is a mail service that uses port 25 to listen for incoming mail using the Simple Mail Transport Protocol (SMTP). Even though you may not specify a port number when using an application, all IP connections involve a connection request to a specific port number using a specific basic protocol

like UDP or TCP.

- most firewalls can determine if a connection request is initiated from outside the firewall or from inside an organization's network. These "stateful" firewalls keep track of sessions and can permit sessions (the flow of traffic back and forth across the firewall) that are only initiated from systems inside the firewall. For example a workstation can request a connection to an outside Web server, but no outside system can initiate a connection to the workstation.
- most firewalls operate at the network layer and are unaware of the actual data or payload carried within network packets. Some firewalls work at the application layer and can examine the data carried within network packets and understand protocols such as HTTP and SMTP. Application-layer firewalls can be configured to only permit valid HTTP and SMTP traffic and exclude everything else.

Now imagine you are paid by an organization to setup a firewall to protect its network. What rules would you put in place? For example:

1. What IP addresses would you allow remote systems to talk to when the remote system (outside your firewall) initiates the connection?
2. When a remote system is allowed to initiate a connection to an IP address what ports on the systems would you allow it to connect to?
3. Are systems within your firewall allowed to make requests to systems outside the firewall?
4. If systems can request connections to outside the firewall are they restricted to certain remote IP addresses?
5. If systems can request connections outside the firewall are they restricted to certain well-known ports such as port 80 for HTTP?
6. Should an application-layer firewall be used to examine data from outside the firewall to insure it conforms to a protocol or format like HTTP, SMTP or HTML?

Given these questions lets see how each decision might effect Flash Communications.

Controlling Connections Initiated from Outside a Firewall

Normally, a firewall will be setup to block connections initiated from outside except where certain servers must be available to the outside world. For example access to port 80 on a Web server (at a certain IP) must be allowed, or if a Web server uses another port such as 8080 it will be allowed. Also port 443 for SSL encrypted Web access may have to be allowed. Similarly access to port 25 will be allowed for the IP address of a mail server.

The only effect of restrictions such as these on Flash communications is if you want to run the Flash Communication Server MX inside a firewall and make it available to the outside world. By default the server accepts connections from Flash movies on port 1935 so port 1935 for the communication server's IP address must be open on the firewall. Illustration 1 shows a successful connection where no firewall is between a remote workstation and a Flash

Communication Server. Illustration 2 shows a deliberately oversimplified situation where a firewall permits access to port 80 but not to port 1935. The simplest solution is to open access through the firewall to port 1935 as in the third illustration.

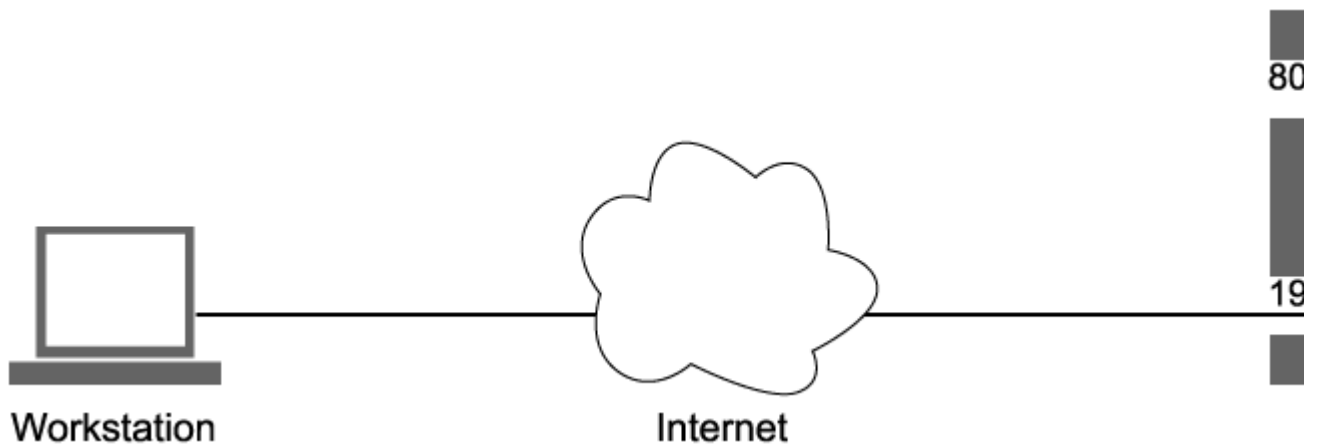


Illustration 1. A direct RTMP connection request to a host running the Flash Communication Server. A Web server is running on the same host as the Communication Server and is available at port 80. The Flash Communication Server MX is available at port 1935.

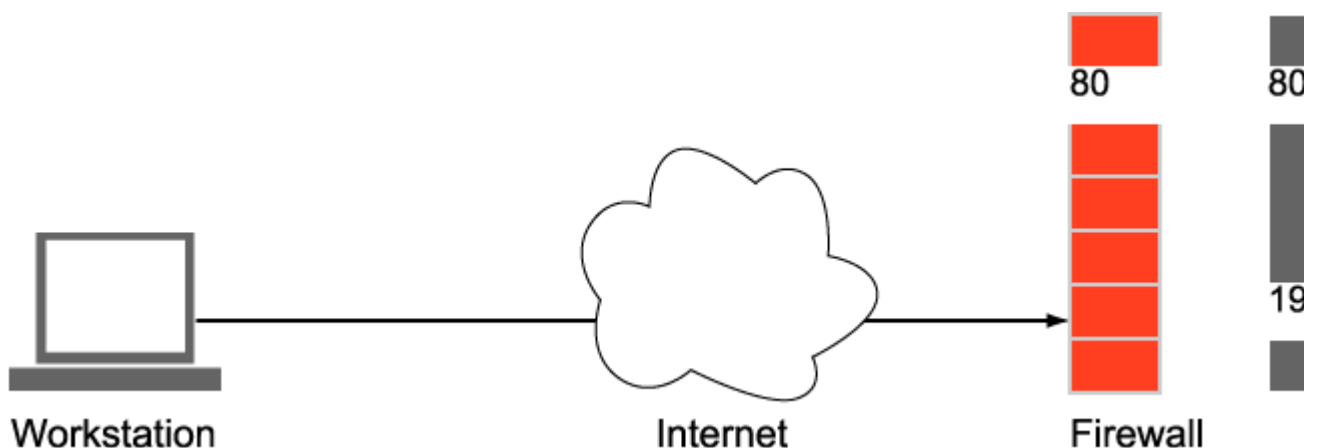


Illustration 2. A connection request blocked by a firewall that does not permit outside requests to port 1935.

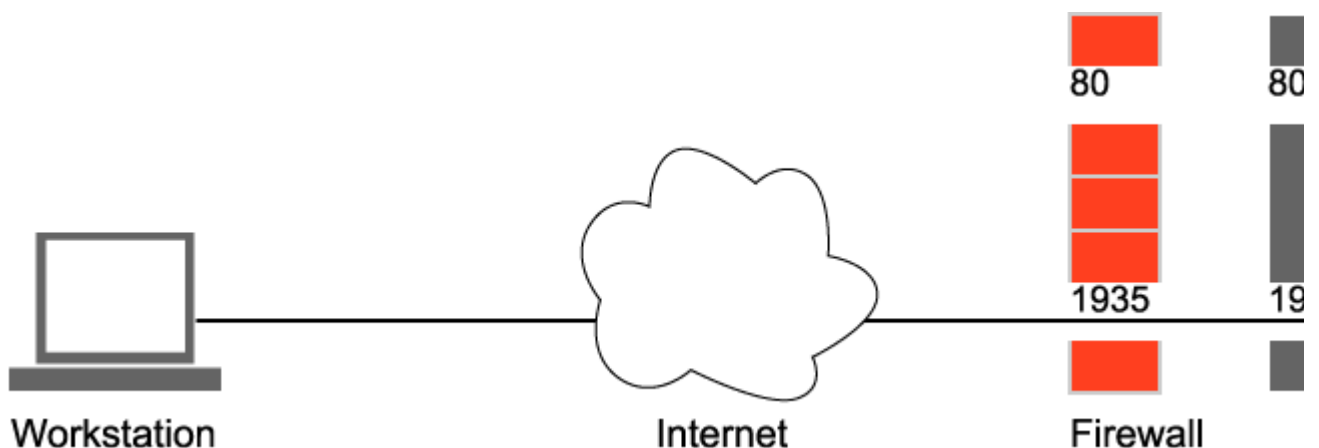


Illustration 3. Opening a "hole" in the firewall to allow access to the Flash Communication Server's default port.

The Flash Communication Server MX can also be configured to accept connections on other ports than 1935 and on more than one port at the same time. The <HostPort> tag in the Adaptors.xml file makes this possible. For example to have the communication server accept connections on ports 1935, 8080, 443, and 80 this host port tag might be used:

```
<HostPort>:1935, 8080, 443, 80</HostPort>
```

If any other service is running on the same server at the same IP address do not configure the communication server to listen on the same port. For example if a Web server is using port 80 and/or 443 then these ports cannot be used by the communication server.

The communication server's administrative server normally runs on port 1111. It is a good idea to leave port 1111 blocked so that an attacker cannot reach it at all. However, administrators and developers working remotely may want access to port 1111 in order to use the administration and app_inspector movies. In this case there is a basic conflict between convenience and security. Hosting services will have to leave port 1111 open and use virtual hosts and logins to control what remote users can do. Institutions that do not provide hosting services may not have to open port 1111. If they must allow remote access to port 1111 the IP addresses that are allowed to reach 1111 can be controlled at the firewall or on the server by using the Server.xml file's <Allow> and <Deny> tags. A better solution for institutions is to install a virtual private network to allow authenticated access through the firewall and otherwise leave port 1111 blocked.

Controlling Connections Initiated from Inside a Firewall

The Flash Communication Server MX never has to attempt to connect to a Flash movie - all connections are initiated from Flash. If a firewall permits workstations to make a connection to any port on any system outside the firewall, there will be no connection problems. For example, the following sequence of events would be permitted:

1. A user's browser requests a Web page containing a SWF file from a remote Web server on port 80.
2. After the swf file is downloaded and played within the browser, it connects to the communication server at port 1935.

In both cases the user's workstation initiated the request so everything will work fine. The firewall will remember that the connection to port 1935 was requested by the user's workstation and data will be allowed to move back and forth between the workstation and the server.

Many organizations restrict the ports local workstation's can request a connection to outside the firewall. Some may restrict access to just the ports

commonly used by Web servers. The most common Web server ports are 80 for HTTP and 443 for HTTPS. Port 8080 is often used as a secondary address for HTTP.

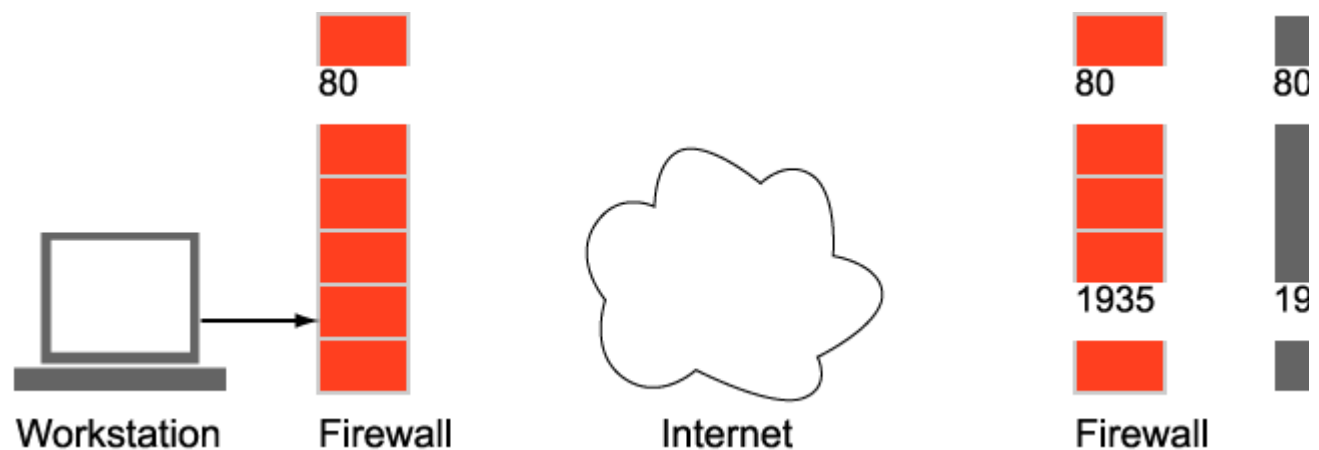


Illustration 4. Client access to the communication server blocked by a firewall that does not permit requests on port 1935.

The simplest solution to the problem of port 1935 being blocked is to setup the communication server to also accept network connections on ports 443 and/or port 80. This assumes no other service is using those ports on the same IP address that your communication server is using - something we'll return to in a moment. To listen on ports 1935, 443, and 80 use a host port tag in the `Adaptors.xml` file something like this:

```
<HostPort>:1935, 443, 80</HostPort>
```

Or, if a specific IP address must be identified for an adaptor:

```
<HostPort>XXX.XXX.XXX.XXX:1935, 443, 80</HostPort>
```

where `xxx.xxx.xxx.xxx` is the IP address.

In many cases this is all you need to get past many firewalls without even using HTTP tunneling. This works because of a feature built into the `NetConnection` object. When you do not specify a port number in an `rtmp` address, Flash will attempt to connect to port 1935. But, if it fails it will then try to connect to port 443, and if that fails, it will try port 80. So no coding is required to access ports 1935, 443, or port 80 if you do not specify a port in the `rtmp` address.

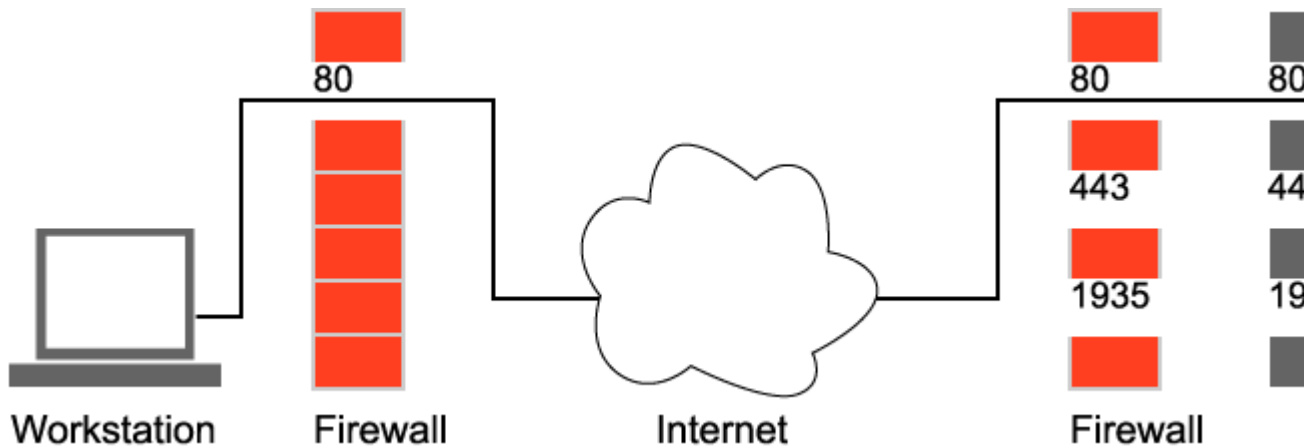


Illustration 5. A successful connection request to the communication server on port 80. Note the Web server shown earlier is no longer using port 80.

Here is a fictitious example of a full URI without a port number:

```
nc.connect("rtmp://host.domain.com/myApp/myInstance");
```

and here is an example of a relative URI without a port number:

```
nc.connect("rtmp:/myApp/myInstance");
```

The relative URI only works if the communication server is available from the same address as the Web server.

Provided the communication server accepts connections on ports 1935, 443, and 80, and the firewall allows the workstation to initiate a connection on one of these ports the connection will succeed. When a Web server is running on the same server as the Flash Communication Server MX port 80 and possibly port 443 may not be available. Here are some solutions to this problem:

1. Don't put a Web server on the same machine that runs the Flash Communication Server! This is probably a good idea anyway but may not always be possible. You don't have to worry about sandbox security restrictions when the Web server is on a different host. A Flash movie can attempt to connect to any Flash Communication server so serving your swf files from a different host is not a problem.
2. Configure your server to answer at two separate IP addresses associated with different host names. You can use two separate network cards to do this or in some cases may even be able to use a single network adapter.
3. Configure your Web server to only accept connections on ports 8080 and possibly 443, leaving port 80 available for the communication server. The problem with this approach is that some firewalls may block access to your web server if they only allow access to port 80.

Another option is to configure the communication server to accept connections on port 8080. There are two drawbacks to this. One is that many firewalls will block port 8080. The other is that you have to explicitly attempt to connect to port 8080 using ActionScript in your Flash movies. For example if the movie

fails to connect to ports 1935, 443, and 80 then the NetConnection's `onStatus()` handler will receive an information object indicating the connection attempt has failed (`NetConnection.Connect.Failed`). At that point you can try to connect again using another port number. Here is an example of a full URI with the port number 8080 included:

```
nc.connect("rtmp://host.domain.com:8080/myApp/myInstance");
```

and here is an example of a relative URI with a port number:

```
nc.connect("rtmp://:8080/myApp/myInstance");
```

When you specify a port number Flash will only attempt to connect using the port number you supply.

A Note on XML Socket Servers

The Flash player treats XML socket connections differently than it does RTMP connections. XML socket connections are only allowed to host ports 1024 or higher. Port 8080 would be permitted but 443 and 80 are not. Also, the player's sandbox restrictions apply to XML socket connections but not to RTMP connections.

Application-Layer Firewalls

If firewalls only allowed or denied connections based on IP addresses, ports, and if the connection was requested from inside the firewall, there would be no need for HTTP tunneling. In simple terms all that would be necessary is to allow the Flash Communication Server to accept connections on port 80. However, application-level firewalls can inspect the data within packets as they travel over the network to see if they conform to the protocol normally associated with a certain port. In the case of port 80 these firewalls will almost certainly look for properly formatted HTTP headers and HTTP transactions. When an RTMP connection is established over port 80 it will not resemble an HTTP transaction. An application-layer firewall will detect this and will deny the connection.

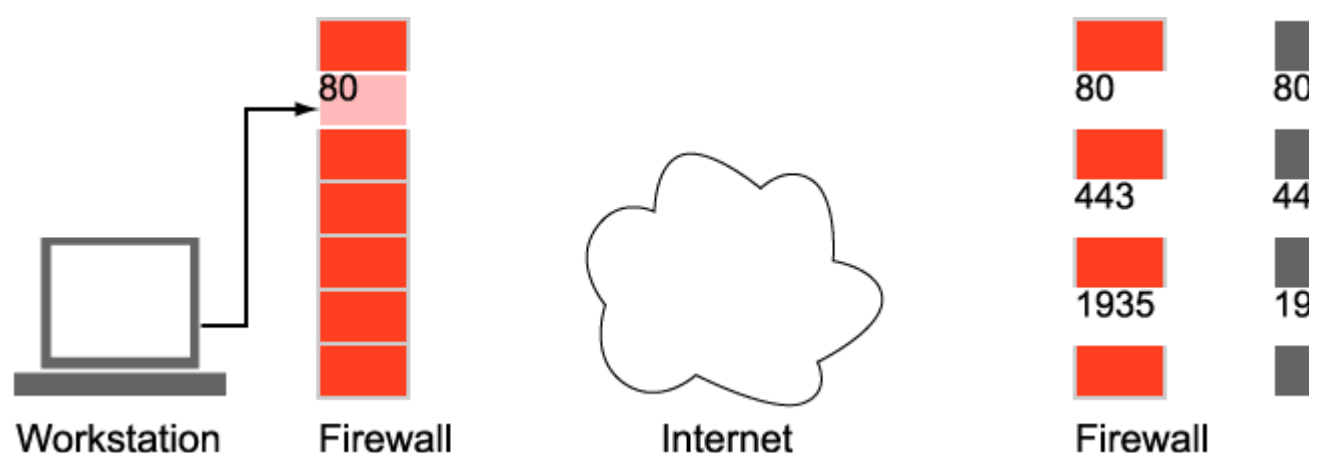


Illustration 6. A firewall that only allows HTTP transactions blocks a request for

an RTMP connection on port 80.

HTTP tunneling can get around this problem. Instead of maintaining a simple RTMP connection the Flash player and Flash Communication Server can exchange RTMP data such as video, audio, and ActionScript data by wrapping the RTMP data inside HTTP formatted requests. For example the Flash player will send a request that begins with a standard HTTP header followed by RTMP data (not shown here).

```
POST /send/20250680/0 HTTP/1.1
User-Agent: Shockwave Flash
Host: myHost.myDomain.com:80
Content-Length: 1537
Connection: Keep-Alive
Cache-Control: no-cache
```

The server will respond with an HTTP header that looks like this followed by additional data.

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Length: 3074
Cache-Control: no-cache
Server: FlashCom/1.5
```

In order to maintain the same type of constant open communications as RTMP provides, the player polls the server constantly with HTTP requests. Polling the communication server with regular requests and wrapping data inside HTTP requests is less efficient than pure RTMP. So, in general you should attempt to use RTMP where ever you can and resort to HTTP tunneling only when you have to.

Fortunately, using HTTP tunneling is not difficult. It is built into version 1.5 of the server and version 6,0,65,0 and later versions of the player. On the server any port that the communication server accepts connections on can be used for either regular RTMP connections or for HTTP. Within the player two features were added. First a new protocol was allowed within connection URIs. For example to explicitly attempt to connect to a server using HTTP tunneling the RTMPT protocol can be specified:

```
nc.connect("rtmpt://host.domain.com/myApp/myInstance");
```

When RTMPT is used in a URI and no port is specified the player will attempt to connect to port 80 using HTTP tunneling. Second, a fourth default behavior was added to how the player attempts to connect when only RTMP is specified and no port is given:

```
nc.connect("rtmp://host.domain.com/myApp/myInstance");
```

As of version 6,0,65,0 of the player the following ports and protocols will be attempted in the order listed. If a connection using the first protocol and port combination fails, then the second one will be tried and so on down the list.

Sequence	Port Number	Protocol
1	1935	RTMP
2	443	RTMP
3	80	RTMP
4	80	RTMPT

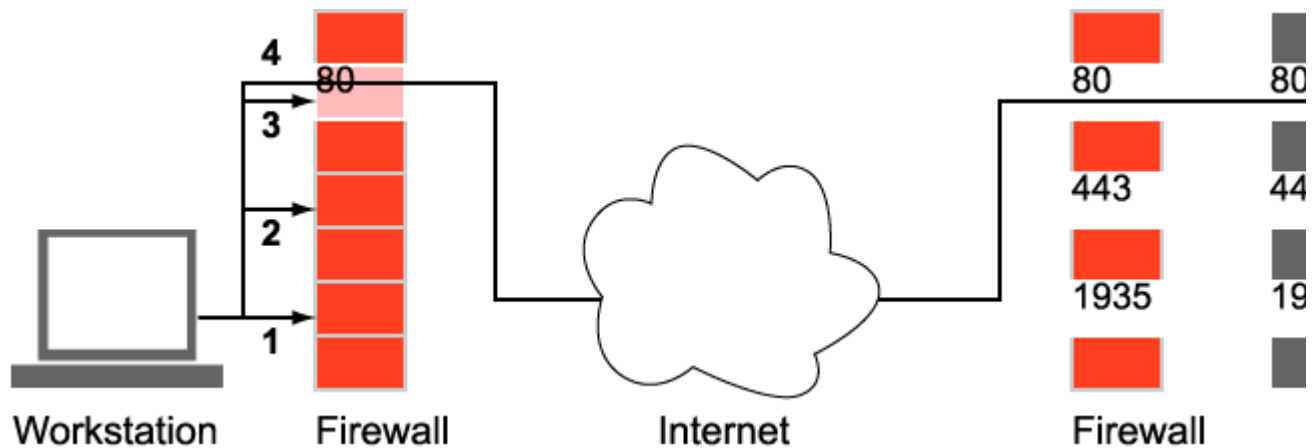


Illustration 7. Four attempts to get through a firewall. Attempts 1, 2, and 3 using the RTMP protocol fail. The fourth attempt succeeds when RTMPT is used on port 80.

The Trouble with Proxy Servers

Another way to make your organization's network even more secure is to use a proxy server as an intermediary between computers inside your organization's network and the Internet. Web proxy servers are often used as caching servers to reduce network utilization. However, proxy servers can often do more than that. Combined with a network-layer firewall, a proxy server makes it possible to completely protect a workstation from directly connecting to anything outside an organization's network while still allowing access to the Web. Instead of requesting a Web page directly from a remote server the browser must request it from the proxy server. The proxy server will look up the page in its cache and return the cached version of the page to the browser. If the page is not in cache the proxy server will request it from the remote server and return it to the browser. With the proxy server acting as an intermediary the firewall can be configured to block all requests - even to port 80 - to remote machines by everything inside the network except the proxy server. See illustration 8.

Before HTTP tunneling, users behind proxy servers that were not allowed to directly connect to the Internet could not connect to a communication server - with tunneling they often can. However, HTTP tunneling does not guarantee success. Proxy servers can do more than act as intermediaries and cache Web pages. They are an easy way to control what external resources can be accessed outside the network. For example a proxy administrator can create a list of sites that are forbidden - effectively denying access to everyone behind

the proxy server to those sites. Some proxy servers can be configured to examine and filter data in greater detail than required to simply act as a Web proxy. For example some can be configured to only accept certain content or MIME types such as text/html. When a Web server returns a Web page to a browser the HTTP header it returns will include a content type. Here's an example header from a Web server returning a Web page:

```
HTTP/1.1 200 OK
Server: Netscape-Enterprise/6.0
Date: Sat, 24 May 2003 01:09:43 GMT
Content-type: text/html
Etag: "c96066c3-1-0-1e8"
Last-modified: Tue, 25 Jun 1996 19:11:18 GMT
Content-length: 488
Accept-ranges: bytes
```

And, here is the header for an image being returned by a Web server:

```
HTTP/1.1 200 OK
Server: Netscape-Enterprise/6.0
Date: Sat, 24 May 2003 01:51:17 GMT
Content-type: image/gif
Etag: "782311ca-126-0-flae"
Last-modified: Mon, 06 May 2002 21:10:38 GMT
Content-length: 61870
Accept-ranges: bytes
```

The data returned by the Communication Server is not text and the headers used by the Communication Server do not include a Content-type statement. So, there are no guarantees that even RTMPT traffic will get through every proxy server. In fact this is no surprise as proxy servers and application-layer firewalls are designed to filter content based on a wide variety of rules setup by the administrator of the system. A security group determined to only allow entry of text/html and some image file formats (for example image/gif) can usually find sophisticated enough tools to succeed. In cases where even RTMPT connections can't get through a proxy server you may have to ask your client to contact the administrator of the organization's firewall and proxy server to see if an exception can be made.

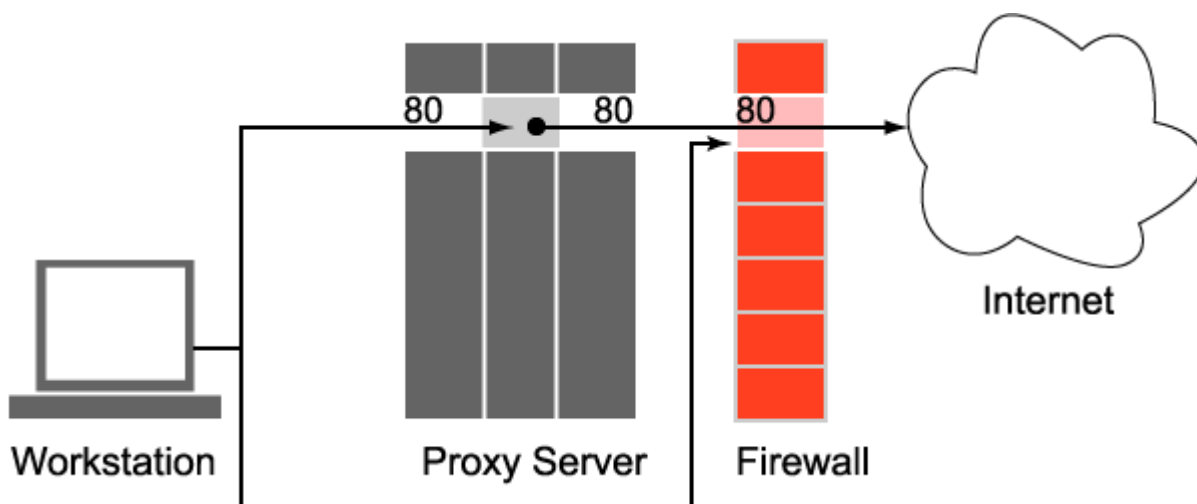


Illustration 8. A proxy server and firewall working together. The workstation

cannot connect directly to the Internet and must instead request a resource through a proxy server. In turn the proxy server fetches the resource from the Internet and passes it back to the workstation. The workstation never actually connects to a system outside the firewall.

Trying More than One Port at Once

Macromedia released updated versions of the communication components with version 1.5 of the Communication Server. One notable update was some new code within the Simple Connect Component. An extra feature was added to speed up connections to servers that cannot be reached at the default RTMP port 1935. If the Application Directory parameter for the SimpleConnect component specifies "rtmp" as the protocol, and does not supply a port number, the Simple Connect component will do the following:

1. Create a NetConnection object and use it to attempt to connect to the address in the normal way beginning with port 1935.
2. Create a second NetConnection object, wait 250 milliseconds, and attempt to connect to the server using RTMPT at port 80.
3. Accept whatever connection succeeds first and close and delete the other NetConnection object.

Strictly speaking this is not necessary. Eventually, the default behavior of the player will be to try ports 1935, 443, and 80 with RTMP and then to try port 80 with RTMPT. However, all these attempts take time. Attempting to connect with RTMPT to port 80 after 250 milliseconds will speed up the connection process dramatically if the first RTMP connection attempt fails. Have a look at the actualConnect() method of the FCSimpleConnectClass to see how Macromedia did this. Of course not everyone uses the Simple Connect component so here is a slightly modified version of Macromedia's code that does pretty much the same thing. It is written to be placed on the main timeline but can be adapted to a more object oriented approach.

```
// The onConnect function is called when a connection is made.
function onConnect(nc){
    _global.main_nc = nc; // Assign the successful nc to a global variable.
    main_nc.onStatus = function(info){
        // your own onStatus handler code goes here
        // to handle errors and other events.
    }
    // Now connect other components and/or initialize your program.
}

// The onConnectFailed function is called if a connection cannot be made.
function onConnectFailed(info){
    // Examine the info object and
    // report why you can't connect.
}

rtmp_nc = new NetConnection();
rtmp_nc.owner = this;
rtmp_nc.onStatus = function(info) {
    this.pending = false;
```

```

    if (info.code == "NetConnection.Connect.Success") {
        if (this.owner.rtmp_nc.pending) {
            rtmp_nc.onStatus = null;
            rtmp_nc.close();
            rtmp_nc = null;
        }
        onConnect(this);
    } else
        if (! rtmp_nc.pending)
            onConnectFailed(info);
}

rtmp_nc = new NetConnection();
rtmp_nc.owner = this;
rtmp_nc.onStatus = function(info) {
    this.pending = false;
    if (info.code == "NetConnection.Connect.Success") {
        if (rtmp_nc.pending) {
            rtmp_nc.onStatus = null;
            rtmp_nc.close();
            rtmp_nc = null;
        }
        onConnect(this);
    } else
        if (! rtmp_nc.pending)
            onConnectFailed(info);
}

rtmp_nc.pending = true;
rtmp_nc.pending = true;

// Try to connect with rtmp right away:
rtmp_nc.connect("rtmp://myHost.myDomain.com/myApp/myInstance", userName, pas

// Setup and interval to try rtmp in 250 milliseconds:
connectionID = setInterval(connectRTMPT, 250);

function connectRTMPT(){
    clearInterval(connectionID);
    rtmp_nc.connect("rtmp://myHost.myDomain.com/myApp/myInstance", userName,
}

```

You can modify this to try other ports. For example if you can't use port 80 and want to try port 8080:

```

function connectRTMPT(){
    clearInterval(connectionID);
    rtmp_nc.connect("rtmp://myHost.myDomain.com:8080/myApp/myInstance", user
}

```

Encrypted Communications!

When run within a browser the Flash player uses the browser's services to communicate via RTMPT. Since browsers also support SSL why not add in support for secure encrypted HTTPS tunneling? That is exactly what Macromedia did. You can request an HTTPS tunnel using a third protocol: RTMPS. For example:

```
nc.connect("rtmps://myHost.myDomain.com/myApp/myInstance");
```

By default the player will attempt to connect to port 443 when rtmps is specified. There is just one problem. The Flash Communication Server MX version 1.5 does not support SSL communications. So, if the player tries to connect with rtmps directly to a communication server the connection will fail. But all is not lost. If you want SSL encrypted communications there are three things you can do:

1. Install wrapper software on your communication server such as stunnel (see <http://www.stunnel.org/>) to handle the encrypted traffic on a port such as 443 and pass unencrypted communications back and forth to the Communication Server on another port such as 1935.
2. Have the Flash movie connect to a second server running software like stunnel. In turn the second server can be configured to connect to the communication server and pass back and forth unencrypted data. In this scenario the system running stunnel acts as an intermediary for all communications. Encrypted communications data passes back and forth between the player and the server running stunnel. The unencrypted communications are passed back and forth between the intermediate host running stunnel and the communication server.
3. Use a dedicated SSL accelerator tied into a network switch to accept encrypted traffic and decrypt communications before they reach the Communication Server.

Of the three methods the third is by far the best and though most expensive solution. Dedicated network/SSL hardware performs better than server-side solutions and in many respects is transparent to the communication server so requires less configuration work. SSL hardware accelerators that work with network switches were designed to entirely remove the load of processing SSL from Web servers. Configuring stunnel can be challenging. For example if you do not setup transparent proxy mode every Flash client will appear to be connecting from the IP address of the stunnel machine. Also, clients that disconnect from stunnel may leave ghost connections to the Communication server that have to be detected and closed. Finally, regardless of what setup you use, if you issue your own certificates consider purchasing one from a recognized authority instead. In our tests we ran into problems with the way Windows handled unrecognized certificate authorities.

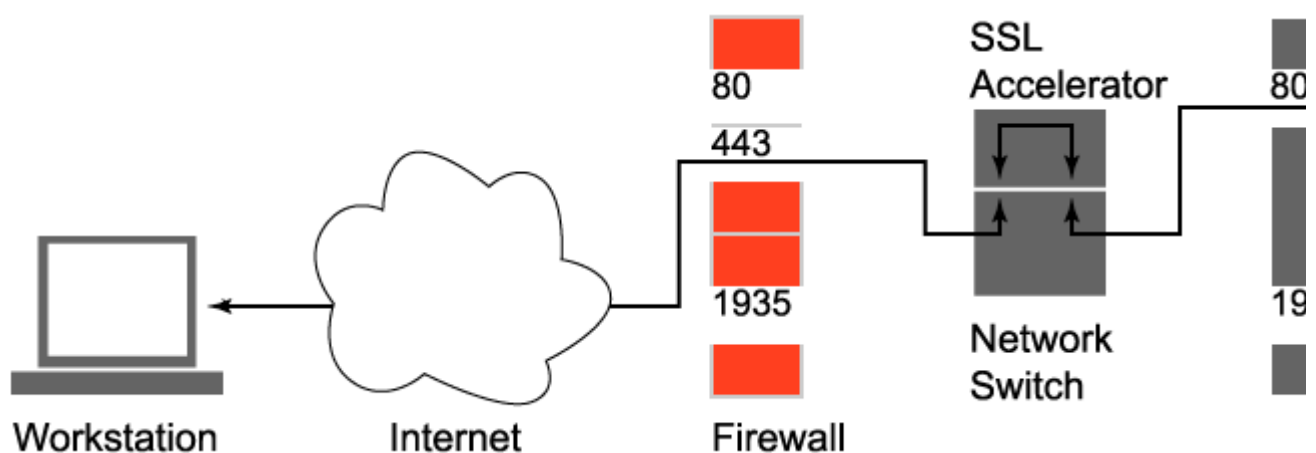


Illustration 9. A network switch and SSL accelerator work together to pass encrypted information back and forth to the workstation and unencrypted information back and forth to the communication server.

Summary

Macromedia has gone a long way to solving connection problems created by firewalls and proxy servers by introducing HTTP tunneling. It is not possible to solve every connection problem created by proxy servers and firewalls because they are often configured with extremely restrictive rules. For good measure Macromedia made it possible to secure all communication between the Flash player running in a browser and the Flash Communication Server. For anyone who wants truly secure real-time audio, video, and data communications that is a great new feature.