

Explore Flash Communication Server MX Webbuilder Las Vegas 9 September 2002 Phillip Kerman

What is it?

A server letting Flash MX authors create movies that share—in real time—data, video, and audio. Anyone with the free Flash Player 6 can view these creations.

What can it do?

Obvious uses include video conferencing and collaborative work applications. For example, an ad agency can create a shared whiteboard to let several people make comments on a proposed layout. In addition to real-time sharing, you can also record data. That means others can view a meeting they missed after the fact... or people can post messages (video, audio, text, graphic) at a convenient time.

There are also lots of immediate uses in education. You can build a teacher-student application that gives the instructor extra powers so they can control what is viewed by each student. For example, a chat application can allow the teacher to chat with each student individually or selectively call on a student to share their question with the other students. Similarly, a live tech-help application can let support personnel conduct simultaneous private sessions with several customers.

The wide range of existing features in FlashCom will give people a chance to come up with all kinds of interesting applications—more than anyone can imagine right now. Keep in mind that although live video is the most exciting feature, simply sharing data in real-time is very powerful—and, it's bandwidth friendly. Add the fact that the user interface is built in Flash (which has a huge user base) and you should see that the potential is great.

See my sample application (and leave a message) at: www.phillipkerman.com/machine

How do you make it do that?

To deliver an application you simply post your .swfs and supporting .html files to a web-server as you do with any Flash application. In addition you need a Windows 2000/XP/NT4 SP6 based web-server running a licensed version of Flash Communication Server MX (UNIX versions will ship later this year). This is where data including video and audio streams are saved as well as server-side scripts and data files to which you don't want to give users direct access (like passwords).

Building a FlashCom application is scripted entirely in Flash MX using the ActionScript language. You can also use a set of "Flash Communications" components to—in minutes—snap together sophisticated FlashCom applications with—literally—no scripting. In addition there's a new set of ActionScript for server-side logic that gets stored in a text file on the server. Many tasks don't require any server-side ActionScript although often a tiny bit of server side ActionScript can go a long way to simplify certain types of projects. For example, in a card game application the script to perform the role of a dealer is best suited for the server as any one client could drop out leaving everyone else stranded.

Components include: "Simple Connect" to ascertain a user's name and ties all other components together; "Chat" for a typical live-chat; "User Color" to give each user's text an identifiable color; "People List" to show who all is currently connected; "Video Conference" to make a multi-user video chat application; and, "Set Bandwidth" to optimize quality and performance by letting users specify their connection speed.

New ActionScript

The new ActionScript on the client side (that is, in Flash) includes the following objects:

NetConnection lets you point to a FlashCom server, define how to handle events received from the server including "onStatus", and finally connect to the server. All other objects then connect to your NetConnection instance.

SharedObject (remote) is similar to local shared objects as data is stored in a file. However, several clients can connect to a single remote SO. Plus, any time a client changes an SO, all connected clients are notified. Also, SOs are one way for clients (or the server itself) to trigger functions in on the server (or in each client). That is, a form of messaging gets layered on top of SOs.

NetStream objects each establish a one-way stream of audio and/or video. For two-way video you simply need two NetStream instances. One client can publish a NetStream (either live or to get recorded) and then another client can make a NetStream instance subscribe to ("play") a named stream published by the other client.

Camera and Microphone Objects let you set properties of the user's webcam (like frame rate, quality, and motion level) or their microphone (such as echo level, silence level, and gain). You can attach a Camera object to a video instance on screen but in order for other clients to see the picture and sound you'll want to attach the Camera and Microphone to a NetStream being published. To view another client's camera you just attach an incoming NetStream (one being played) to a video instance.

Additionally, Server Side ActionScript (which, I suppose, is all new) has a couple additional objects as well as a few objects that share the same name with a client-side object.

Application objects containing information about a particular application instance. For example, in the Application object you can store variables such as a list of connected users. Because you can have multiple instances of a single application (think "chat rooms") each one maintains information about a single instance.

Client objects refer to individual logged in users. You can ascertain (from the server side) the IP address of a particular user and alternatively deny them access for example. Also, it's easy to write functions that get shared among all client instances.

A different way to think

Unlike traditional ActionScripting, you can't execute functions that involve a server and expect an immediate response. Additionally, Flash can't pause while waiting. Before you execute such a script you must identify how Flash is to use the reply and *then* execute the command. For all clients connected to a particular SharedObject, there are automatic triggers for when anyone changes a value in the SharedObject. Each client can define how to respond to such onStatus or onResult events. In addition, you can call custom events directly to fulfill messaging needs. No matter how you trigger things, you always have to define how your application will respond to events before those event occur. There are also a host of new "what if" considerations. Like, what if two users attempt the same maneuver at about the same time? What if one user closes the browser window? And so on. (By the way, there are ways to overcome all such issues.)

You'll also want to learn a new—albeit very consistent—way of thinking when it comes to status and error messages. The way you determine what's going on is through the onStatus callback. Quite a few standard situations will trigger onStatus events. For example, when the server disconnects a user or one of the other clients changes a value in a SharedObject the onStatus event gets triggered. In order to ascertain the exact reason an onStatus event happened you need to sort through the "information object" that is received as a parameter. Macromedia has designed a mechanism to pass detailed information through a single parameter (instead of a variable number of parameters). These information objects are just generic Flash objects (also called "associative arrays"). Each info object has a `code` property and a `level` property. You just need to extract the value for the property about which you're interested. For example, if you want a particular script to trigger if when the user gets connected you check to see if the code property has the value "NetConnection.Connect.Success" like this:

```
my_nc.onStatus=function (info){
    if(info.code == "NetConnection.Connect.Success"){
        //you're connected, so run this script...
    }
}
```

At first, the info object may be difficult to understand, but it's quite consistent. Like many of the new ways of thinking required for FlashCom, it's really pretty easy once you adapt.

How much does it cost?

Personal Edition (maximum 10 simultaneous connections and 1 Mbit/second): \$499
(Additional Personal Editions—up to 5—will increase limits linearly.)

Professional Edition (maximum 500 simultaneous connections and 10 Mbit/second): \$4500
(Additional "expansion packs" add 500/10 to the Pro version only and cost \$4000 each.)

You also need Flash MX (\$499 or \$199 upgrade) to author FlashCom applications.

Additionally, a range of host options are available from www.mediatemple.net

Where can you learn more?

A presentation showing Macromedia's positioning of the entire MX product line:
www.macromedia.com/software/mx/presentation/

A great article exploring scenarios and applications for FlashCom:
www.macromedia.com/desdev/mx/flashcom/articles/comserver.pdf

Jumping off point for technical information including security:
www.macromedia.com/desdev/mx/flashcom/

General product information:
www.macromedia.com/software/flashcom/