

Eyeball Video Suite 3.6

Eyeball SDK Overview



Eyeball Video Suite 3.6

Eyeball SDK Overview

Table of Contents

1. Introduction	3
2. How Applications and Services Based on Eyeball SDK Work.....	4
3. Using the Eyeball SDK	5
3.1 Creating ActiveX Objects.....	6
3.2 Using the Features and Functionalities.....	6
3.2.1 Using the Instant Messaging Control	6
3.2.2 Using the Video Session Control	7
3.2.3 Using the Chat Room Control	9
3.2.4 Using the Whiteboard Control.....	10
3.2.5 Using the Video Message Control.....	10
3.2.6 Using the Video Mail Control	11
3.3 Handling Event Notifications	12
4. Sample Applications.....	14
4.1 Sample code - E-Commerce Web Site	14
4.2 Sample code - Live Video Meeting.....	15
4.3 Sample code - Web Chat Room	16
5. Legal and Contact Information.....	18

Last Modified: October 27, 2003

Copyright © 2001-2003 Eyeball Networks Inc. All rights reserved. Patents pending.

1. Introduction

The Eyeball Software Development Kit (SDK) gives application developers the tools needed to integrate live video communications features into new or existing Internet-based applications and services. Developers can use Eyeball SDK to create a custom client application with one-to-one or group video communications, text messaging, whiteboarding, video broadcasting and video mailboxes. These video-enabled applications communicate with server components in Eyeball Video Suite to seamlessly deliver interactive, high-quality video to users. Eyeball SDK incorporates Eyeball's patent-pending Any-Bandwidth™ and Any-Firewall™ Technologies to ensure that deploying Internet video communications to a group of users is both easy and seamless.

The Eyeball SDK provides a powerful solution for developers to integrate the following features into their products quickly and easily:

- Interactive one-to-one and group video communications,
- Instant text messaging, online presence detection and contact list management,
- File transfer,
- Video mailbox,
- Whiteboarding and presentation sharing,
- Live video broadcasting, and
- Interactive chat rooms

Applications and services that can benefit from these features include on-line customer support, Web communities, distributed games, distance education, and entertainment.

Some of the benefits of Eyeball SDK include:

- *Guaranteed Best Video Quality.* There are several challenges that affect Internet video quality, including unpredictable bandwidth, packet loss, latency, jitter, and CPU heterogeneity. Eyeball SDK uses Eyeball's patent-pending Any-Bandwidth™ Technology to guarantee the best possible video quality over any Internet connection and desktop.
- *Seamless Firewall Interoperability.* Firewalls and modems can inadvertently block video sessions, resulting in frustration, lost productivity, and missed revenue opportunities. Eyeball's patent-pending Any-Firewall™ Technology ensures seamless video delivery between any combination of standard firewalls and modems without compromising firewall security.
- *Scalable Peer-to-Peer Architecture.* Sending audio and video data through a central relay server consumes costly server hardware and bandwidth resources and reduces video quality. Eyeball Video Suite is based on a peer-to-peer architecture in which data is sent directly from one client computer to another. As a result Eyeball Video Suite scales to millions of users with minimal operational costs and maximum video quality.
- *Language-Independent Development:* Development support for C, C++, Javascript, Visual Basic, and VBScript programming languages.
- *Embedded Video Support.* Eyeball SDK allows application developers to implement video communications as either a standalone client or embedded into a web application. When embedded, the video client is sent to a user seamlessly as part of a web page. This eliminates the need for users to install a separate video client and allows developers to update their application without the installation of new video client software.

The Eyeball SDK consists of:

- Eyeball SDK ActiveX Library,
- Eyeball SDK Programmers' Reference, and
- Source code for sample applications.

For a detailed description of Eyeball SDK, please refer to Eyeball SDK Programmers' Reference.

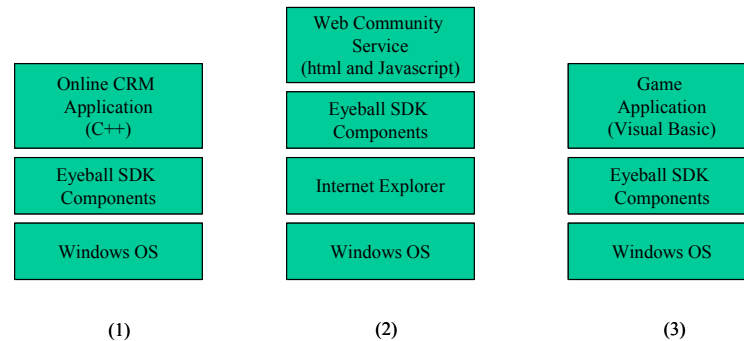


Figure 1: The Eyeball SDK allows development of stand-alone and web-based applications and services using languages such as (1) C++, (2) html and Javascript and (3) Visual Basic.

Figure 1 shows how developers can build stand-alone/web-based applications of their choice by using Eyeball SDK. The rest of the document is organized as follows. Section 2 depicts how applications and services developed on Eyeball SDK work. Section 3 describes how to use the Eyeball SDK. This section explains how to create the ActiveX control objects, how to use the features and functionalities of the SDK and how to handle the notification events. Section 4 presents sample code for an E-commerce web site, a simple video meeting application and a Web chat room.

2. How Applications and Services Based on Eyeball SDK Work

Figure 2 shows how applications and services developed using Eyeball SDK work with active support from Eyeball servers such as Eyeball Video Communications Server, Eyeball Video Broadcast Server, Eyeball Video Mail Server. Software developers can develop new applications, services or websites that will have Eyeball SDK components embedded in them. The figure illustrates two kinds of client applications: stand-alone applications that execute in Microsoft Windows operating systems and web-based applications and services that can be accessed using a web browser such as Microsoft Internet Explorer.

In order for these applications and services to provide interactive video communication capabilities, the embedded Eyeball SDK components need to communicate with the one or more Eyeball servers running with valid licenses from Eyeball Networks Inc. Eyeball SDK comes with a development server license that is valid for testing the developed applications and services. However, proper production licenses of appropriate Eyeball servers are required before the developed applications and/or services may be used for reasons other than testing.

Eyeball servers need to be configured properly for some of the Eyeball SDK features to function properly. For example, in order to provide a web-based service using Eyeball SDK, the URL of

the web-service needs to be added to the authorized URLs lists of the related servers. For further details on server licensing and configuration issues, please refer to the documentation available in the Eyeball Video Suite.

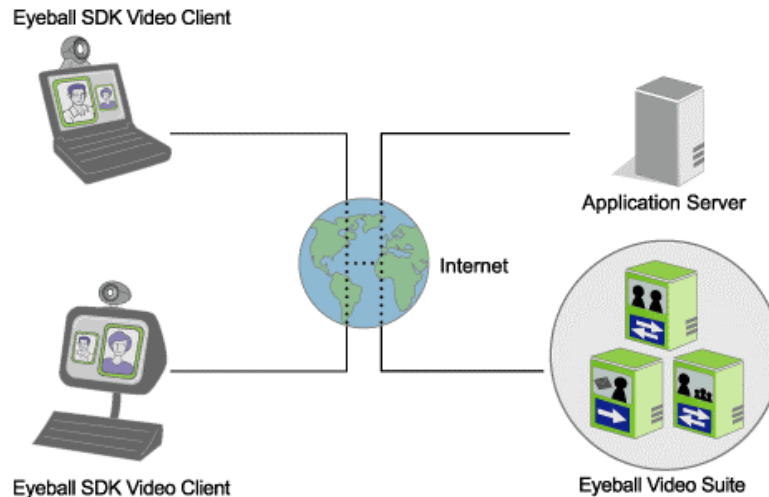


Figure 2: Applications and services based on Eyeball SDK works with Eyeball Video Communications Server, Eyeball Video Broadcast Server and Eyeball Video Mail Server.

3. Using the Eyeball SDK

The Eyeball SDK consists of a library of ActiveX controls that provides programmers a high-level interface to the main features and functionalities of Eyeball Communicator. Currently Eyeball SDK consists of 6 main ActiveX controls:

1. *Instant Messaging control*: Supports contact list, presence detection, instant text messaging and file transfer.
2. *Video Session control*: Supports session creation and joining, session configuration and tool management, floor control, instant messaging, media settings, device selection and volume adjustment.
3. *Video Message control*: Supports recording and playing of video messages including changing of capture devices.
4. *Chat Room control*: Supports interactive text chat rooms including join and leave, room categories and creation of private and public rooms.
5. *Whiteboard control*: Supports shared whiteboarding including a rich set of drawing tools and editing functions and sharing of Microsoft PowerPoint presentations and Windows Bitmap files.
6. *Video Mail control*: Supports web-based video mailbox service including sending and viewing of video mails and user-created folders.

In this section, we present the Eyeball SDK from a web-programmers' point of view. However, programmers using other languages such as C++ and Visual Basic will also get a clear understanding of the supported features and functionalities.

3.1 Creating ActiveX Objects

The following HTML code shows how to embed some ActiveX control objects into a web page:

```
<OBJECT
  id="InstantMessagingCtl"
  classid="CLSID:D3D0E5D1-0FBE-4DAC-886A-F86912390EBF">
</OBJECT>

<OBJECT
  id="VideoSessionCtl"
  classid="CLSID:4B9A05DC-9CE9-4259-95E8-90C6E581101B">
</OBJECT>

<OBJECT
  id="VideoWindowCtl"
  classid="CLSID:CA06EE71-7348-44c4-9540-AAF0E6BD1515">
  <param name="IsWndless" value=true>
</OBJECT>

<OBJECT
  id="ChatRoomCtl"
  classid="CLSID:866CD2CC-5C58-448E-AEB1-6EE473717B1D">
</OBJECT>

<OBJECT
  id="WhiteboardCtl"
  classid="CLSID:ADC9C5DF-27FD-4637-B2C5-4221414B5754">
</OBJECT>

<OBJECT
  id="VideoMessageCtl"
  classid="CLSID: 02CA9974-B6AC-497E-A371-73580432B0F6">
</OBJECT>

<OBJECT
  id="VideoMailCtl"
  classid="CLSID:3FF30BA7-7A56-426C-9AD2-1F0A270DC305">
</OBJECT>
```

3.2 Using the Features and Functionalities

Some of the main features and functionalities of the Eyeball SDK are described below.

3.2.1 Using the Instant Messaging Control

Contact List

The control allows contacts to be programmatically added and deleted. The following code adds a new user to the Contact List:

```
InstantMessagingCtl.AddContact("UserID");
```

and the following code deletes a user from the Contact List:

```
InstantMessagingCtl.DeleteContact("UserID");
```

The following code detects a user's status:

```
sStatus = InstantMessagingCtl.GetUserStatus(
    "UserID");
```

A copy of the Contact List can be generated to support operations such as printing or iteration. The following code retrieves a copy of the current Contact List, represented as a VB array:

```
aContactList = InstantMessagingCtl.GetContactList();
```

Instant Messaging

To send a text message to another online user use:

```
InstantMessagingCtl.SendTextMessage(
    "UserID", "hello");
```

File Transfer

The following code sends a file to another online user:

```
FileId = InstantMessagingCtl.SendFile(
    "UserID", "testfile.txt");
```

However, when the control is embedded in a web page, the file name should be left empty. This will show a standard dialog window for the user to select the file to be sent.

To accept a file from another user use the following code:

```
InstantMessagingCtl.AcceptFileTransfer(
    FileId, "testfile.txt", true);
```

Multiple files can be sent and received simultaneously.

3.2.2 Using the Video Session Control

This control supports both peer-to-peer and server-based audio/video data-transport. The control defines four user types:

- **Host:** Creator of the session. The host enjoys the highest control over a session such as inviting others, changing user types and changing session permissions and configurations. Host becomes the default presenter.
- **Presenter:** A user that sends both audio and video by default and has limited control over a session such as changing the user type of speakers and audience.

- **Speaker:** A user that sends audio and video by default, but has no control over the session.
- **Audience:** A user that does not send audio or video by default.

At any time, a session will have a host, zero or one presenter, and zero or more speakers and audience.

Session Configuration

Before creating a session, the host can define a few configuration parameters for a session such as the bit-rates of video and audio streams, and capabilities and permissions of each user types.

For example, the following will disable audience to send video or audio:

```
VideoSessionCtl.UserStream("audience", "video") = "";
VideoSessionCtl.UserStream("audience", "audio") = "";
```

And this will enable speakers to send video and audio:

```
VideoSessionCtl.UserStream("speaker", "video")
    = "10000 50000";
VideoSessionCtl.UserStream("speaker", "audio")
    = "melp gsm";
```

This allows speakers to send two video streams, one stream with 10 kbps and the other stream with 50 kbps, and two audio streams, one using MELP encoding and the other using GSM encoding.

To enable speakers to send video and audio to everyone use:

```
VideoSessionCtl.StreamTarget("speaker") = "everyone";
```

Or to enable speakers to send video and audio to the host and presenter only use:

```
VideoSessionCtl.StreamTarget("speaker") = "controller";
```

Meeting Session

A new session can be started using:

```
VideoSessionCtl.CreateSession(
    SessionName, SessionPassword);
```

To join an existing meeting session use:

```
VideoSessionCtl.JoinSession(SessionName, SessionPassword);
```

The following code ends the session:

```
VideoSessionCtl.EndSession();
```

To invite a user to join the session, the host can use the following:

```
VideoSessionCtl.CallUser(UserID);
```

Floor Control

The audience and speakers can send request to host or presenter for changing their user types. The host can change the user type of any other participant (presenter, speaker or audience). The presenter can change the user type of audience and speakers.

For example, suppose user H is currently the session host, and user X and Y are both audience. Host H can use the following to make X a presenter:

```
VideoSessionCtl.ChangeUserType("X", "presenter");
```

User Y at any time can request to be a speaker, presenter or audience as follows:

```
VideoSessionCtl.RequestUserType("speaker");
```

After X becomes a presenter, X can make Y a speaker as follows:

```
VideoSessionCtl.ChangeUserType("Y", "speaker");
```

Instant Messaging

VideoSessionCtl provides a method for sending instant messages among the attendees of session. To send a message to everyone in the session use:

```
VideoSessionCtl.SendTextMessage("", "How is it going?");
```

3.2.3 Using the Chat Room Control

ChatRoomCtl handles all the Eyeball interactive meeting room features. To get the room list of a certain category use the following code:

```
ChatRoomCtl.Category = "Movie";  
RoomNamesArray = ChatRoomCtl.GetRoomList();
```

To enter a room use:

```
ChatRoomCtl.EnterRoom("RoomName", "RoomPassword");
```

User can get the current room users list by:

```
aUserList = ChatRoomCtl.GetUserList();
```

The following sends a text message to the room:

```
ChatRoomCtl.SendTextChatMsg("hello");
```

To leave the current room use:

```
ChatRoomCtl.LeaveRoom();
```

To create a meeting room, use the following code:

```
ChatRoomCtl.CreateRoom("RoomName", iMaxLimit, iRoomType,  
    "RoomGreeting", "RoomDetail", "RoomPassword");
```

User needs to specify the maximum number of people allowed in the room by the integer parameter `iMaxLimit` and public/private room by `iRoomType` parameter.

3.2.4 Using the Whiteboard Control

The Whiteboard control supports shared whiteboarding between two or more users. Users can load Microsoft PowerPoint presentations and Windows Bitmap files, and/or edit the whiteboard using a rich set of tools such as pen, line, rectangle, ellipse and text.

Sharing a Presentation

To share a Microsoft PowerPoint file with the users in the session use:

```
WhiteboardCtl.OpenFile("MyPowerPointFile.ppt");
```

Drawing Graphics

`WhiteboardCtl` provides a set of methods to draw shape, line, rectangle, curve, highlighter, etc. on the whiteboard. For example, to draw a rectangle use:

```
WhiteboardCtl.Tool = "rectangle";
```

The following code selects the pen tool:

```
WhiteboardCtl.Tool = "pen";
```

The user can now use the mouse as a pen to draw freehand curves on the whiteboard. The created shapes are sent to the connected whiteboards.

3.2.5 Using the Video Message Control

Video Message control allows to record and play standard AVI files. These AVI files can be used in file transfer for sending video messages or in Video Mail control for sending video mail.

Record a Message

To record a video message use:

```
VideoMessageCtl.StartRecord();
```

And to stop recording:

```
VideoMessageCtl.StopRecord();
```

To send this recorded message to an online user, use the file transfer service provided by InstantMessagingCtl:

```
sMessageName = VideoMessageCtl.RecordedMessageName();  
fileId = InstantMessagingCtl.SendVideoMessage(  
    "UserID", "hello", sMessageName);
```

Play a Message

To play the received video message from the remote contact:

```
sFileName = InstantMessagingCtl.GetTransferFile(fileID);  
VideoMessageCtl.PlayedMessageName = sFileName;  
VideoMessageCtl.StartPlay();
```

3.2.6 Using the Video Mail Control

Video Mail control allows to upload generic files or AVI files created by Video Message control to Video Mail Daemon. Video messages can be downloaded from Video Mail Daemon and played back by Video Message control. It also support message folder management.

Send a Video Mail

To send a 10-second video message clip (e.g., "hello.avi") to the remote user A use:

```
VideoMailCtl.SendVideoMessage("hello.avi", "A",  
    "Subject", "TextMessage",  
    10, TRUE/*bSaveOutgoingMail*/);
```

Note that the video message is stored in the server and user A will get notified when A logs into the Eyeball Video Mail Server.

Download and Play a Video Mail

To retrieve a mail list in a specified folder use:

```
aList = VideoMailCtl.GetMessageList("folderName");
```

To download a message from the retrieved mail list use:

```
VideoMailCtl.DownloadMessage(iMessageID);
```

Where iMessageID is an element of aList.

Once the video message is downloaded completely, an event is fired to notify the user that this message is ready to play.

```

<SCRIPT language=JScript for=VideoMailCtl
event=OnFileReceived(sFileName)>
    VideoMailCtl_OnFileReceived(sFileName);
</SCRIPT>

function VideoMailCtl_OnFileReceived(sFileName)
{
    VideoMessageCtl.PlayedMessageName=sFileName;
    VideoMessageCtl.StartPlay();
}

```

Folder Management

The control provides a set of methods to enable end users to manage the mail folders. To create a folder use:

```
VideoMailCtl.CreateFolder("FolderName");
```

To delete a folder use:

```
VideoMailCtl.DeleteFolder("FolderName");
```

To move a message to a specified folder use:

```
VideoMailCtl.MoveMessage(iMessageID, "NewFolderName");
```

3.3 Handling Event Notifications

Most of the ActiveX controls send several event notifications to the Eyeball SDK applications. The programmers need to handle these events as necessary. For example, `InstantMessagingCtl` will notify events such as:

- Text messages,
- File transfer requests, and
- Contact list updates.

Most of the events get fired with necessary information in their parameters. For example, the following code handles the `OnTextMessage` event.

```

<SCRIPT language=JScript for=InstantMessagingCtl
event=OnTextMessage(aMessage)>
    InstantMessagingCtl_OnTextMessage(aMessage);
</SCRIPT>

function InstantMessagingCtl_OnTextMessage(aMessage)
{
    var aMsg = aMessage.toArray();
    var sSender = aMsg[0];
    var sText = aMsg[1];
    var sService = aMsg[2]; // EYEBALL
}

```

```
    alert(sSender + ": " + sText + "; Service:" + sService);
}
```

VideoSessionCtl will notify events like:

- Any user joining/leaving the session,
- Text messages from any user in the session,
- Change of user type and request for user type,
- Incoming call and call response, and
- Request for or release of video window.

As an example, the following code can handle OnUserJoinNotify event from VideoSessionCtl:

```
<SCRIPT language=JScript for= VideoSessionCtl event=
OnUserJoinNotify(aUserInfo)>
    return VideoSessionCtl_OnUserJoinNotify(aUserInfo)
</SCRIPT>
```

```
function VideoSessionCtl_OnUserJoinNotify(aUserInfo)
{
    var aInfo = aUserInfo.toArray();
    var sUserName = aInfo[0];
    var sUserType = aInfo[1];
    alert("User:" + sUserName + " joined the session.");
}
```

The following code can handle OnVideoWindowOpen event from VideoSessionCtl:

```
<SCRIPT language=JScript for= VideoSessionCtl event=
OnVideoWindowOpen(aUserInfo)>
    return VideoSessionCtl_OnVideoWindowOpen(aUserInfo)
</SCRIPT>
```

```
function VideoSessionCtl_OnVideoWindowOpen(aUserInfo)
{
    var aInfo = aUserInfo.toArray();
    var sUserName = aInfo[0];
    var sUserType = aInfo[1];
    hWnd = VideoWindowCtl.GetVideoWindow();
    VideoSessionCtl.AttachVideoWindow(hWnd);
}
```

ChatRoomCtl will notify events like:

- Any user entering/leaving the room,
- Text messages from any user in the room, and
- Status update for any room member.

As an example, the following code can handle OnTextChatMsg event from ChatRoomCtl:

```
<SCRIPT language=JScript for=ChatRoomCtl
event=OnTextChatMsg(msg)>
    return ChatRoomCtl_OnTextChatMsg(msg)
</SCRIPT>

function ChatRoomCtl_OnTextChatMsg(msg)
{
    var aMsg = msg.toArray();
    Alert("Sender: " + aMsg[0] + ":" + aMsg[1]);
}
```

Customer Support Page

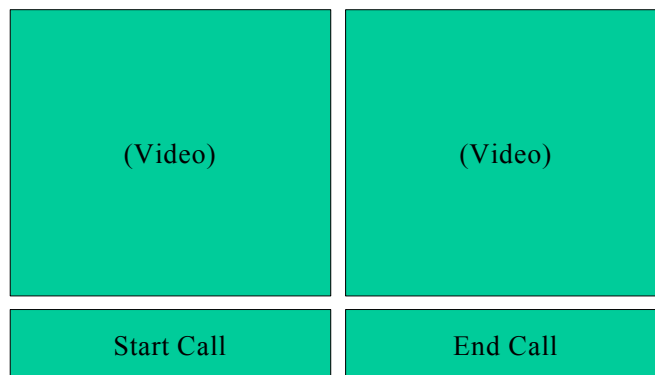


Figure 3: Interface for a customer support web page using the Eyeball SDK.

4. Sample Applications

In this section, we present three sample web applications using the Eyeball SDK:

4.1 Sample code - E-Commerce Web Site

The following example shows how easy it is to integrate the Eyeball SDK into a web site. The demo application implements a one-click connection to a company's customer representative.

Suppose that on the "Contact us" page of the company's web site, there is an `VideoSessionCtl` control and two buttons: one to start the video call, and the other to end the call. The user can click the start button and instantly begin chatting with the customer representative, who is using an Eyeball Web Communicator client.

```
<INPUT type=text name=TextUserID>
```

```

<INPUT type=password name=TextPassword>
<INPUT type=button value="Start Call" name=BtnStart>
<INPUT type=button value="End Call" name=BtnEnd>

// Handle a click on the "Start Call" button
function BtnStart_OnClick()
{
    InstantMessagingCtl.Login(
        TextUserID.value, TextPassword.value);
    VideoSessionCtl.Login(
        TextUserID.value, TextPassword.value);
    VideoSessionCtl.CallUser("CustomerCare");
}

// Handle a click on the "End Call" button
function BtnEnd_OnClick()
{
    VideoSessionCtl.EndSession();
    VideoSessionCtl.Logout();
    InstantMessagingCtl.Logout();
}

```

4.2 Sample code - Live Video Meeting

Here is some sample code showing how a simple video meeting can be implemented:

```

<INPUT type=text name=TextUserID>
<INPUT type=password name=TextPassword>
<INPUT type=text name=TextRemoteID>
<INPUT type=button value="Login" name=BtnLogin>
<INPUT type=button value="Logout" name=BtnLogout>
<INPUT type=text name=TextSessionName>
<INPUT type=password name=TextSessionPassword>
<INPUT type=button value="Create Session" name=BtnCreateSession>
<INPUT type=button value="Join Session" name=BtnJoinSession>
<INPUT type=button value="End Session" name=BtnEndSession>
<INPUT type=button value="Call User" name=BtnCallUser>

// Handle a call request event
function VideoSessionCtl_OnUserCallNotify(sCaller)
{
    var bAccept = confirm("Accept call from " + sCaller + "?");
    VideoSessionCtl.AcceptCall(sCaller, bAccept);
}

// Handle a click on the "Login" button
function BtnLogin_OnClick()
{
    InstantMessagingCtl.Login(
        TextUserID.value, TextPassword.value);
    VideoSessionCtl.Login(

```

```

        TextUserID.value, TextPassword.value);
    }

    // Handle a click on the "Logout" button
    function BtnLogout_OnClick()
    {
        VideoSessionCtl.Logout();
        InstantMessagingCtl.Logout();
    }

    // Handle a click on the "Create Session" button
    function BtnCreateSession_OnClick()
    {
        VideoSessionCtl.CreateSession(
            TextSessionName.value, TextSessionPassword);
    }

    // Handle a click on the "Join Session" button
    function BtnJoinSession_OnClick()
    {
        VideoSessionCtl.JoinSession(
            TextSessionName.value, TextSessionPassword.value);
    }

    // Handle a click on the "End Session" button
    function BtnEndSession_OnClick()
    {
        VideoSessionCtl.EndSession();
    }

    // Handle a click on the "Call User" button
    function BtnCallUser_OnClick()
    {
        VideoSessionCtl.CallUser(TextRemoteID.value);
    }

```

4.3 Sample code - Web Chat Room

Below is the sample code showing how a web-based interactive chat room can be implemented:

```

<INPUT type=text value="" name=TextUserID>
<INPUT type=password value="" name=TextPassword>
<INPUT type=text name=TextMessage>
<INPUT type=button value="Send" name=BtnSend>
<INPUT type=button value="Connect" name=BtnConnect>
<INPUT type=button value="Disconnect" name=BtnDisconnect>
<INPUT type=text value="" name=TextRoomName>
<INPUT type=button value="EnterRoom" name=BtnEnter>
<INPUT type=button value="LeaveRoom" name=BtnLeave >

// handle text message event from other users

```

```

function ChatRoomCtl_OnTextChatMsg(msg)
{
    var aMsg = msg.toArray();
    var sUserID = aMsg[0];
    var sMsg = aMsg[1];
    Alert(sUserID + ":" + sMsg);
}

// handle a click on "Connect" button
function BtnConnect_OnClick()
{
    ChatRoomCtl.Connect(
        TextUserID.value, TextPassword.value);
}

// handle a click on "Disconnect" button
function BtnDisconnect_OnClick()
{
    ChatRoomCtl.Disconnect();
}

// handle a click on "EnterRoom" button
function BtnEnterRoom_OnClick()
{
    ChatRoomCtl.RoomListType = EYEBALL_ROOM;
    ChatRoomCtl.EnterRoom(TextRoomName.value, "");
}

// handle a click on "LeaveRoom" button
function BtnLeaveRoom_OnClick()
{
    ChatRoomCtl.LeaveRoom();
}

// handle a click on "Send" button
function BtnSend_OnClick()
{
    ChatRoomCtl.SendTextChatMsg(TextMessage.value);
}

```

5. Legal and Contact Information

Copyright © 2001-2003 Eyeball Networks Inc. All rights reserved. Patents pending.

Eyeball, Eyeball.com, its logos, Any-Bandwidth™ and Any-Firewall™ are trademarks of Eyeball Networks Inc. All other referenced companies and product names may or may not be trademarks of their respective owners.

For more information visit Eyeball Networks at www.eyeball.com.

Department	Eyeball Chat ID	E-mail
Technical Support	TechSupport	techsupport@eyeball.com
Sales	Sales	sales@eyeball.com
General	Info	info@eyeball.com

Corporate Headquarters:

500-100 Park Royal
West Vancouver, BC V7T 1A2
Tel. 604.921.5993
Fax 604.921.5909