

# Adding Video to Applications with the Oracle Video Server

*An Oracle White Paper*

*March 1998*

# Adding Video to Applications with the Oracle Video Server

## INTRODUCTION

Streaming video technology is proving its usefulness for everything from training programs to product promotions. Oracle provides a number of ways for client applications to play video streamed by an Oracle Video Server. Such flexibility lets people view video content on a wide variety of devices. Personal computers, network computers, kiosks, and set-top boxes can all display video from an Oracle Video Server. Viewers enjoy full-motion, full-screen video with stereo sound.

Oracle provides development tools to quickly add interactive video playback to client applications. These tools include no-programming-required components such as video windows, interactive VCR-style controls (play, pause, stop, volume, rewind, fast forward), popup control menus, and information displays to speed application development. The same tools can also be used to build highly-customized client applications with unique user interfaces.

- The Oracle Video Web Plug-in allows HTML web pages to display video.
- The Oracle Video ActiveX Control allows applications developed with tools such as Visual Basic, Developer 2000, and HTML to integrate video.
- The Oracle Video Java Library allows developers to use video in native Java applications.
- For systems developers, the Oracle Video Server APIs provide even more control over the services of the Oracle Video Server itself.

This document focuses on the ease of client application development. We'll start with an overview of digital video and the Oracle Video Server.

## WHY USE DIGITAL VIDEO

Digital video has become a standard data type that can easily be integrated into applications. Server-based digital video allows video content to be more current, manageable, easily distributed, and cost effective than video distributed on analog tapes or CD-ROMs. Networked digital video, also known as streaming video, uses existing networks to provide live or stored video simultaneously to multiple users.

Today's businesses need to build productivity by enhancing communications and learning. On-demand video is a key technology for increasing efficiency and adding capabilities in these business areas:

- Executive-level and general corporate communications
- Marketing and sales communications
- On-the-job training and learning
- Electronic commerce
- Public information delivery
- Broadcast news
- Entertainment
- Video-based evidence
- On-line video archives

### THE ORACLE VIDEO SERVER

Oracle Video Server (OVS) is a software-based server that stores and manages digital video content. It provides on-demand streaming of digital video over networks to multiple clients. OVS, optionally combined with Oracle databases, enables you to develop and deploy innovative multimedia applications for today's leading-edge businesses.

A core component of Oracle's universal data server, OVS is uniquely designed to store, manage, and stream digital video from standard server platforms to client devices (see Figure 1).

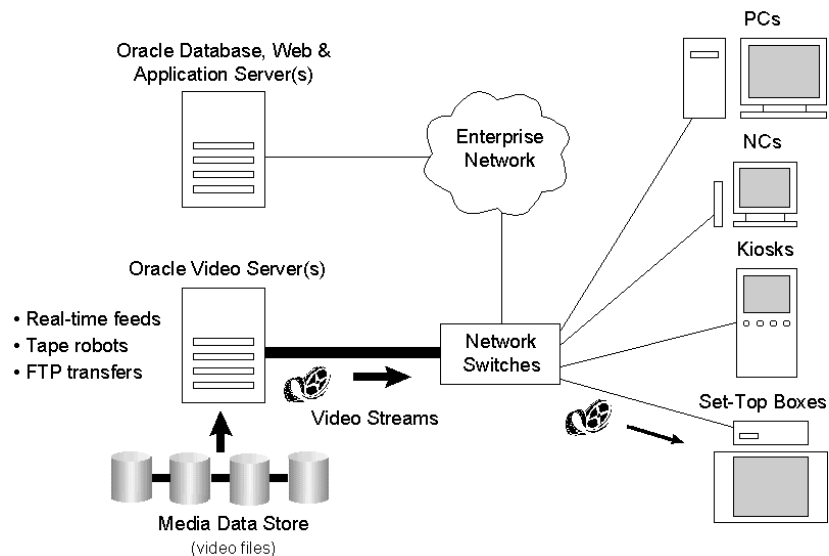


Figure 1 - Oracle Video Server System

OVS streams encoded and compressed digital video to clients, where the streams are decompressed, decoded, and displayed in real-time under interactive control of the application. No video data needs to be stored on the client. Digital video is optimized on the OVS for streaming and sharing by multiple users. Additionally, the OVS supports real-time feeds from video encoders to enable streaming of live events. Robotic tape systems can be used to extend on-line video storage.

Typical system solutions include various database, web, and application servers to manage metadata associated with the video content and to provide additional data management and processing for comprehensive applications. With support for many server platforms (including Intel/WinNT, Sun, DEC, HP, SGI, nCUBE, and others) and many network configurations, the OVS system can be configured to accommodate many deployment requirements.

OVS supports standard IP networks such as Fast Ethernet, FDDI, and IP/ATM. Each network segment can support multiple clients by multiplexing video streams. An OVS can be scaled to support hundreds of concurrent users depending on the hardware platform, network type, and their expandability.

## **VIDEO CONTENT**

The OVS supports multiple video compression/decompression algorithms and delivers content at the appropriate rate for the algorithm. These encoding formats include MPEG-1, MPEG-2, AVI, ClearVideo, Cinepak, and Indeo. OVS also supports audio-only WAV and Voxware streams. A digital video file is associated with a companion “tag” file that contains basic video file information and indexing data.

For example, with MPEG-1, video streaming rates range up to 3 Mbps (million bits per second) for full-screen, full-motion video with CD-quality stereo audio. With MPEG-1, video can be stored with resolutions up to 352 x 240 pixels at 30 frames per second with 24-bit color and CD-quality stereo audio. MPEG-1 encoding typically provides compression ratios of 100:1 to 200:1, resulting in stream rates from 1 to 2 Mbps. Using a 100 Mbps FDDI server segment, OVS can support approximately 40 concurrent users at 2 Mbps per stream.

A video encoded at 2 Mbps requires approximately 900 MB (million bytes) of storage for each hour of content; the advantage of server-based, shared video content is obvious. During playback, an MPEG-1 video stream is decompressed and decoded at the client device. A hardware decoder card may be used, depending on the capabilities of the client device.

MPEG-1 can also be encoded for a smaller window size and lower video and audio quality to reduce stream rates to a few hundred Kbps.

During on-screen video display, the application program can scale the video window using the client machine’s graphics accelerator capabilities. Video codecs other than MPEG are typically decoded only by software.

## CLIENT APPLICATION TOOLS FOR ORACLE VIDEO SERVER

The OVS package includes the Oracle Video Client software, which provides multiple application tools for implementing client applications that operate with the Oracle Video Server.

Oracle Video Client includes both high-level components and application tools to facilitate easy integration with the Oracle Video Server and low-level tools for system developers creating highly customized applications. Oracle Video Client includes a number of preprogrammed components that share a common look-and-feel and a number of APIs:

- Preprogrammed video window
- Preprogrammed VCR-style on-screen controls for video playback
- Preprogrammed popup control menus and information displays
- APIs for video window height, width, and borders
- APIs for video file selection and retrieving video information
- APIs for play-from and play-to positions
- APIs for play, pause, resume, stop, and volume

Oracle Video Client provides the following application tools:

<b>Application Tools</b>	<b>Development Environments</b>
Oracle Video Web Plug-in	HTML pages using Netscape Navigator and Microsoft Internet Explorer; can be controlled by JavaScript and Java applets.
Oracle Video ActiveX Control	ActiveX-compliant environments such as Microsoft Visual Basic, Oracle Developer 2000 (formerly Oracle Forms), and third-party authoring tools; usable in HTML pages with VBScript and JScript and ActiveX-compliant browsers.
Oracle Video Java Library	Native Java development and runtime environments

*Table 1 - Oracle Video Client Application Tools*

These application tools use a common set of components and APIs that handle the complexities of networking with the Oracle Video Server and of managing video and audio streams. They support Microsoft Windows 95 and Windows NT clients and use the standard Microsoft ActiveMovie decoders and graphics facilities.

### ORACLE VIDEO WEB PLUG-IN

The Oracle Video Web Plug-in can be used with both Netscape Navigator and Microsoft Internet Explorer to allow HTML web pages to interactively play video streams from the OVS.

HTML pages use the plug-in by including an <embed> tag. The video appears directly in the web page. The <embed> tag can include a number of attributes to specify the window size, video file, autostart, start and end points, volume, and more. Attributes can also enable VCR-style controls and a popup control menu.

If enabled, users can click the left mouse button to play or pause the video and the right mouse button to open the popup menu. This popup menu contains commands for rewinding, forwarding, setting automatic starting and looping, and showing tool tips and video information.

This plug-in supports the Netscape LiveConnect interface and provides methods that allow JavaScript and Java applications to use the plug-in dynamically and provide customized user interfaces.

### Using the <EMBED> Tag

The <embed> tag for the Oracle Video Web Plug-in supports the following attributes that control the video window and playback:

Attribute	Description
autostart	Specify if video starts when plug-in is loaded
background	Specify a background color behind plug-in area
border	Specify border width around video window
controls	Specify if plug-in displays with on-screen controls
controlmask	Specify which on-screen controls appear with the plug-in
height	Specify height of plug-in, including on-screen controls and border width
hidden	Specify if plug-in hidden or not
leftclick	Specify if left mouse button for play, pause
loop	Specify if video is to loop at end of stream
mediafile	Specify the protocol, server, and video file to play; use except where “src” attribute is used
name	Specify local plug-in name; used by JavaScript and Java applets
playfrom	Specify start point in the video
playto	Specify stop point in the video
popupmenu	Specify if popup control menu to be displayed
sliderrate	Specify time increments for position slider
src	Specify source file and MIME type of video file
tooltips	Specify if tool tips are displayed
type	Specify MIME type of video file, a fixed value; if not used, “src” must be used
volume	Specify volume level
width	Specify width of the plug-in, include border width

Table 2 - Oracle Video Web Plug-in EMBED Tag Attributes

The only required attributes are “autostart”, “height”, “width”, and specification of the video file using either the “mediafile”, “src”, or “type” attribute.

## Oracle Video Web Plug-in with HTML <EMBED> — Sample 1

Figure 2 shows an HTML page with a video window that runs automatically with no on-screen controls. The video window size (352 x 240 pixels) is the standard encoding size for MPEG-1 encoded files.

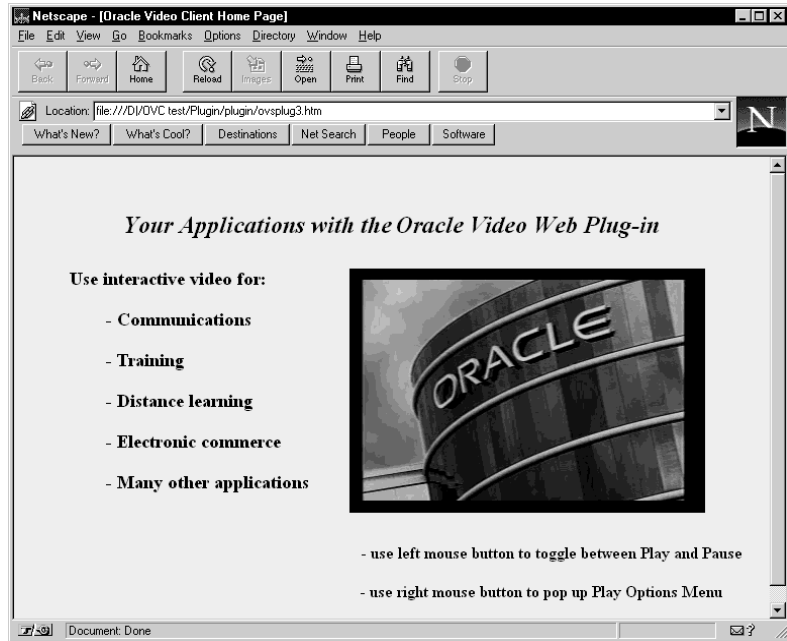


Figure 2 - Oracle Video Web Plug-in Without On-screen Controls

The <embed> tag to include and play this video is simply:

```
<embed type=application/oracle-video width=352 height=240 popupmenu=true  
leftclick=true  
mediafile=/mds/video/  
yourvideofile  
autostart=true>
```

While no visible controls are shown, the mouse and popup menu can be used to control the video. The left mouse button toggles between play and pause. The right mouse button opens the popup menu shown in Figure 3.

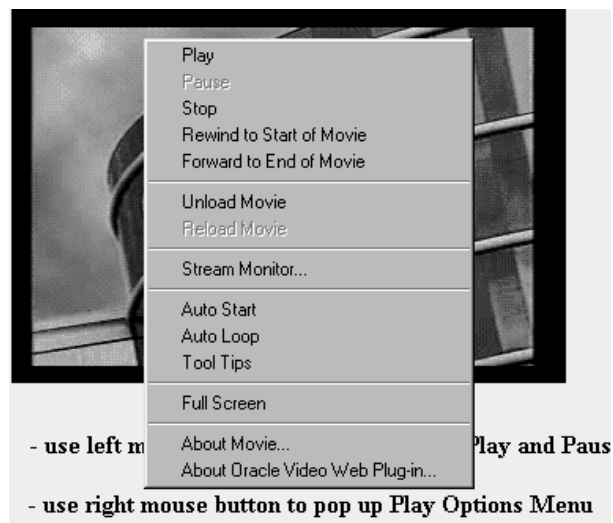


Figure 3 - Oracle Video Web Plug-in with Popup Control Menu

## Oracle Video Web Plug-in with HTML <EMBED> — Sample 2

Figure 4 shows a large HTML page with a video window that includes the optional, VCR-style controls. These controls allow the user to play, pause, stop, move to a different point in the video, adjust the volume, and see the current playback mode and time from the start of the video.



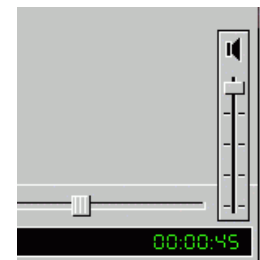
*Figure 4 - Oracle Video Web Plug-in with On-screen Controls*

To scale the 240x352 pixel video by 150%, the <embed> tag uses a width of 352 x 150% and a height of 240 x 150% plus the height of the on-screen controls (45 pixels).

The <embed> tag to include and play this video is:

```
<embed type=application/oracle-video width=528 height=405 controls=true  
controlmask=controller+statusline mediafile=/m/s/video/yourvideo.mpi  
autostart=true >
```

Clicking the speaker icon in the lower right corner of the video window displays a slider the user can adjust to control the volume.



*Figure 5 - Volume Control Slider*

## ORACLE VIDEO WEB PLUG-IN WITH JAVASCRIPT AND JAVA APPLETS

The Oracle Video Client also includes two Java classes that allow JavaScript and Java applets to control the Oracle Video Web Plug-in. These classes are OviPlayer and OviObserver. In addition to the <embed> tag attributes, these Java classes support a number of methods, including methods to control the video stream and detect use of the VCR-style controls.

### Java Class Methods

Method	Description
advise()	Specify OviObserver object
forward()	Forward video stream to end position
getLength()	Get total length of video stream
getMaxPos()	Get maximum stream position
getMinPos()	Get minimum stream position
getObserver()	Get OviObserver object
getPos()	Get current stream position
getState()	Get current state of player
getVol()	Get play volume
load()	Load video file specified
pause()	Pause video stream
play()	Play from beginning or play-from position
resume()	Resume play of video stream
rewind()	Rewind to beginning or play-from position
setAutoStart()	Set autostart
setFullScreen()	Set playback in full-screen mode
setLoop()	Set looping
setPopupMenu()	Set popup control menu
setPos()	Set video stream to new position
setVol()	Set volume
stop()	Stop, rewind to beginning or play-from position
unload()	Unload current video stream

*Table 3 - Methods of OviPlayer Class*

Method	Description
onPositionChange()	Notify applet when video stream changes position
onStop()	Notify applet at end of video stream

*Table 4 - Methods of OviObserver Class*

### Oracle Video Web Plug-in with JavaScript — Sample 3

Figure 6 shows an HTML page with play, pause, and close buttons controlled by a short section of JavaScript code. In order to control the plug-in with JavaScript, the web browser must support the Netscape LiveConnect interface.



Figure 6 - Oracle Video Web Plug-in with Java Controls

#### JavaScript Code for Oracle Video Web Plug-in Control

The JavaScript code to control the Oracle Video Web Plug-in is:

```
<SCRIPT language="JavaScript">
function play() {
    // if player is currently in state realized --> call play
    if (document.OVSplugin.getState() == 4) {
        if (document.OVSplugin.play())
            return true;
    }
    // if player is in any other state --> call resume
    else {
        if (document.OVSplugin.resume())
            return true;
    }
    return false;
}
</SCRIPT>
<!-- Embed the Oracle Video Client Web Plug-in -->
<embed name=OVSplugin border=0 width=320 height=240
type=application/oracle-video autostart=false controls=false
controlmask="" playfrom=beginning playto=end loop=true
leftclick=true tooltips=false popupMenu=true volume=100
mediafile="vstcp://mds/yourvideofile.mpi">
</embed>
```

```

<P>
<center>
<form name=form1>
<input type=button onclick="play()" value=Play>
<input type=button onclick="OVSpugin.pause()" value=Pause>
<input type=button onclick="document.OVSpugin.unload()"
value=Close>
</form>

```

#### Oracle Video Web Plug-in with Java Applet — Sample 4

The Play, Pause, and Close buttons in Figure 6 could also be controlled with a Java applet that uses the Oracle Video Java Library. In order to control the plug-in with a Java applet, the Web browser must support the Netscape LiveConnect interface.

#### HTML to Embed Java applet for Oracle Video Web Plug-in

The code to use a Java applet to control the Oracle Video Web Plug-in is:

```

<!-- Embed the Oracle Video Client Web Plug-in -->
<embed name=OVSpugin border=0 width=320 height=240
type=application/oracle-video autostart=false controls=false
controlmask="" playfrom=beginning playto=end loop=true
leftclick=true tooltips=false popupMenu=true volume=100
mediafile="vstcp:///mds/video/ovs_mpg1_2048k.mpi">
</embed>
<P>
 
<!-- The 'MAYSCRIPT="true"' flag in the applet tag is
critical. This tells the browser's security manager to allow
communication between the applet and the plug-in. The value of
the applet parameter, pluginName, must agree with the 'name'
attribute of the embed object (the plug-in). In this case the
plug-in name attribute has the value 'OVSpugin', and the below
applet parameter, pluginName, also has the value 'OVSpugin'.
This agreement of attribute / parameter values is imperative
for the communication between the plug-in and the Java applet.
-->
<!-- Embed the Java Applet Controls -->
<applet
  code=simpleJava.class
  name=OVC_Applet
  MAYSCRIPT="true"
  width=320
  height=30>
  <param name=pluginName value=OVSpugin>
</applet>

```

## Java applet for Oracle Video Web Plug-in

The Java applet code is:

```
//
// simpleJava.java:    Applet
//
import java.applet.*;
import java.awt.*;
//provide access to objects
import netscape.javascript.JSObject;
import netscape.javascript.JSException;
import OviObserver;    //plug-in listener (not implemented)
import OviPlayer;     //OviPlayer interface
//
// Main Class for applet simpleJava
//
public class simpleJava extends Applet
{
    // Members for applet parameters
    // <type> <MemberVar> = <Default Value>
    private String m_plgname = "";
    //GUI Objects
    Button btnPlay      = null;
    Button btnPause     = null;
    Button btnClose     = null;
    JSObject document, window;
    OviPlayer ovi       = null;

    // The init() method is called by the AWT when an applet is
    // first loaded or reloaded. Override this method to perform
    // whatever initialization your applet needs, such as
    // initializing data structures, loading images or fonts,
    // creating frame windows, setting the layout manager, or
    // adding UI components.
    public void init()
    {
        // Parameter support
        // The following code retrieves the value of the
        // applet parameter
        String param;
        // param: Parameter description
        param = getParameter("pluginName");
        if (param != null)
            m_plgname = param;
        resize(320, 30);
        setBackground(Color.white);
        // Place additional initialization code here
        btnPlay      = new Button ("Play");
        btnPause     = new Button ("Pause");
        btnClose     = new Button ("Close");
        add (btnPlay);
        add (btnPause);
        add (btnClose);
        // Get JavaScript object (document)
        // Retrieve the document object from the Browser
        // environment
    }
}
```

```

public void start()
{
}
public void stop()
{
}
public void destroy()
{
    /* Clean Up Pointers */
    ovi          = null;
    document     = null;
    window       = null;
    System.gc();
}
// Mouse support:
// The mouseUp() method is called if the mouse button is
// released while the mouse cursor is over the applet's
// portion of the screen.
public boolean mouseUp(Event evt, int x, int y) {
    // Grab Plug-in object
    // Retrieve the plug-in object handle from the browser
    // context
    ovi = getPlugin();
    boolean cond = false;
    if(evt.target == btnPlay) {
        if (ovi.getState() == ovi.ST_REALIZED)
            ovi.play();
        else
            ovi.resume();
        cond = true;
    }
    if (evt.target == btnPause) {
        ovi.pause();
        cond = true;
    }
    if (evt.target == btnClose) {
        ovi.unload();
        cond = true;
    }
    return cond;
}
// Get JS object handles and pointer to plugin. Retrieves
// JavaScript Object contexts from Browser. This function
// depends upon the Netscape supplied Java package,
// netscape.javascript. This function will not work under
// Internet Explorer.
public OviPlayer getPlugin() {
    OviPlayer plugin = null;
    ovi              = null;
    document         = null;
    window           = null;
    System.gc ();
    try {
        window = JSObject.getWindow(this); //grab window obj
    }
    catch(NullPointerException e) {
        System.out.println("getWindow() failed in
getPlugin()");
    }
    if (window != null)

```

```

        try {
            //grab document obj
            document = (JSObject)
window.getMember("document");
        }
        catch(NullPointerException e) {
            System.out.println("getMember() failed in
getPlugin()");
        }
        if (document != null) {
            try {
                plugin = (OviPlayer)
document.getMember(m_plgname);
            }
            catch(NullPointerException e) {
                System.out.println("unable to grab plugin");
            }
        }
        else {
            System.out.println("Document null in
getPlugin()\n");
        }
        return plugin;
    } //end getPlugin
}

```

### ORACLE VIDEO ACTIVE X CONTROL

The Oracle Video ActiveX Control allows developers to easily integrate streaming digital video using ActiveX-compliant application tools. The Oracle Video ActiveX Control can be used with such ActiveX-compliant development tools as Visual Basic, Visual C++, Oracle Developer 2000, and many third-party multimedia authoring tools. ActiveX is a Microsoft technology for the Windows environment.

The Oracle Video ActiveX Control includes methods, properties, and events for controlling video in applications that access the Oracle Video Server. The Oracle Video ActiveX Control can also be used in an HTML document and controlled using VBScript and JScript. Using VBScript with Microsoft Internet Explorer provides the greatest amount of control. VBScript provides a subset of Visual Basic that can use all the control's methods, properties, and events.

#### ActiveX Control Methods, Properties, and Events

The Oracle Video ActiveX Control provides the following methods, properties, and events:

Method	Description
Forward	Forward video to end or position set by PlayTo property
GetInfo	Return information array for current video
GetPos	Return current stream position as hh:mm:ss or frames from beginning
GetStats	Return information array of current play statistics
GetVol	Return current volume setting

<b>Method</b>	<b>Description</b>
ImportFileAs	Activate dialog box to select video on local disk
ImportStreamAs	Activate dialog box to select video on Oracle Video Server
Load	Request video server allocate resources and loads specified video
Pause	Pause playback at current position
Play	Start playback from current position
Resume	Resume playback from pause state
Rewind	Rewind video to beginning or position set by PlayFrom property
SetPos	Set video position as hh:mm:ss or frames from beginning
SetVol	Set the volume of the video (0 to 100)
ShowInfoDialog	Activate display box showing information on current video
ShowStatsDialog	Activate display box showing play statistics on current video
Stop	Stop video and reposition at beginning or position set by PlayFrom
Unload	Requests video server to free resources for current video

*Table 5 - Methods of Oracle Video ActiveX Control*

In addition to the standard control properties provided by application tools themselves, the Oracle Video ActiveX Control provides the following properties to control the video window and video playback. Most properties can be both specified and returned as a parameter.

<b>Property</b>	<b>Description</b>
AutoStart	Specify if video starts automatically
BorderStyle	Specify border style
EnableLeftClick	Specify if left mouse button for play, pause
Enable Popup	Specify if popup control menu
Height	Specify height of OVCAX1 object
IsLoaded	Video loaded - 0 for false, 1 for true
Left	Specify left position of OVCAX1 object
Loop	Specify if video is to loop at end of stream
MediaFile	Specify the video file to play
PlayFrom	Specify start point in the video
PlayTo	Specify stop point in the video
ShowControls	Specify if on-screen controls displayed
ShowPositionAndStatus	Specify if on-screen status line displayed
State	Playback state indicator, read-only
TimerFrequency	Specify time increment for position slider
Top	Specify top position of OVCAX1 object
Width	Specify width of OVCAX1 object

*Table 6 - Properties of Oracle Video ActiveX Control*

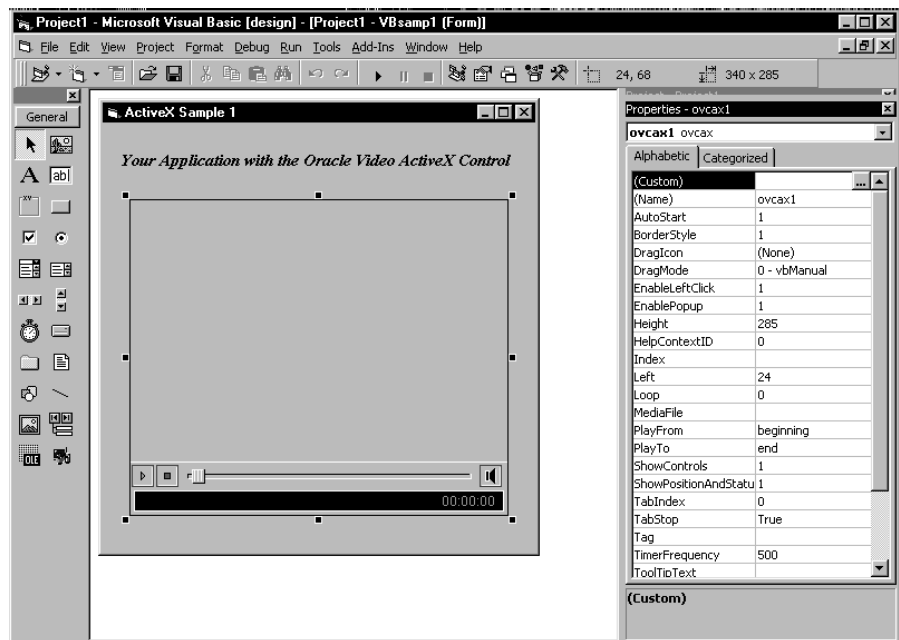
Events are messages the Oracle Video ActiveX Control sends to an application. The application can perform actions in response to an event. The following events are provided by the Oracle Video ActiveX Control.

Event	Description
Completed	Occurs when video stream is completed
LeftClick	Occurs when left-mouse button clicked over video window
PlayStarted	Occurs when video is started via Play method
Resumed	Occurs when video is resumed via Resume method
RightClick	Occurs when right-mouse button clicked over video window
Stopped	Occurs when video is stopped via Stop method

*Table 7 - Events of Oracle Video ActiveX Control*

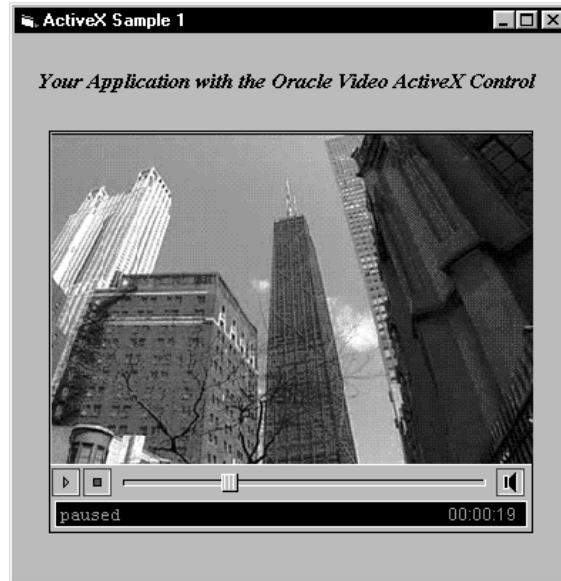
### Oracle Video ActiveX Control — Sample 1

Figure 7 shows the Visual Basic development environment for a video window with the on-screen controls. The left mouse button can be used for play and pause and the right mouse button activates a popup menu for other controls and playback information. Developers set the relevant Oracle Video ActiveX Control properties using the Visual Basic property window shown to the right.



*Figure 7 - Visual Basic with Oracle Video ActiveX Control (OVCAx1)*

Figure 8 shows the same window at run-time.



*Figure 8 - ActiveX Control Sample 1 with Integrated On-screen Controls*

The code to include this sample is simply:

```
'  
' Oracle Video ActiveX Control Sample 1  
'  
' - OVC window with ShowControls true  
'  
  
Private Sub Form_Load()  
    Dim MediaFile As String  
    'video file spec set in your app  
    MediaFile = "/mds/yourvideofile.mpi"  
    ovcax1.Load (MediaFile)  
End Sub
```

Clicking the right-mouse button on the video window opens a popup menu like the one for the Oracle Video Web Plug-in (Figure 3).

## Oracle Video ActiveX Control — Sample 2

Figure 9 shows an application in which the default on-screen controls have been replaced by buttons added by the application developer. The Oracle Video ActiveX Control can easily be customized to match the look-and-feel of custom applications.

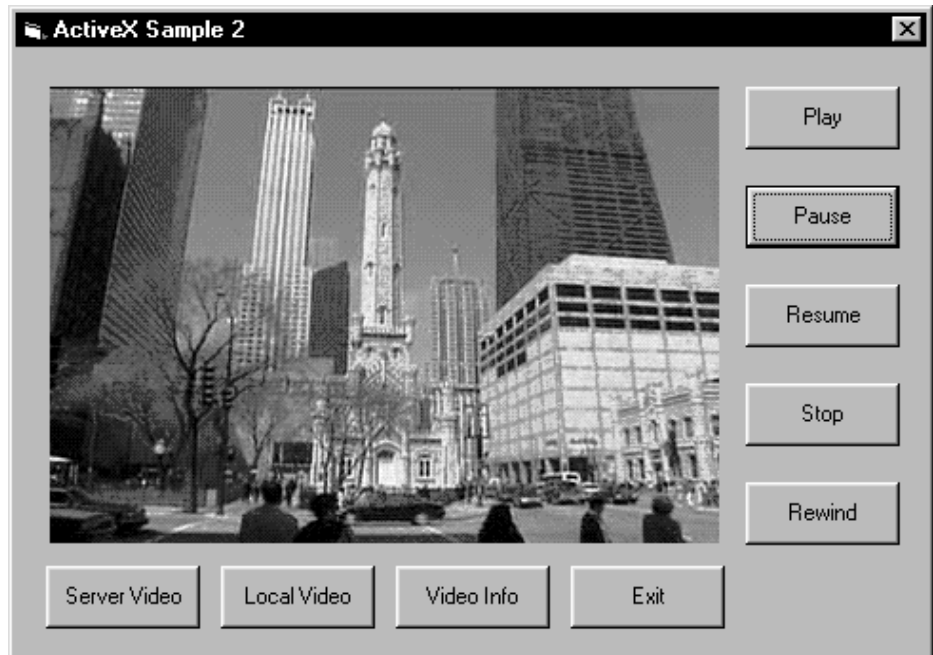


Figure 9 - ActiveX Control Sample 2 App with Control Options

Clicking the “Server Video” button opens a preprogrammed dialog that lists videos available on the Oracle Video Server. In the code example, see the “Private Sub btnGetServer\_Click” routine.

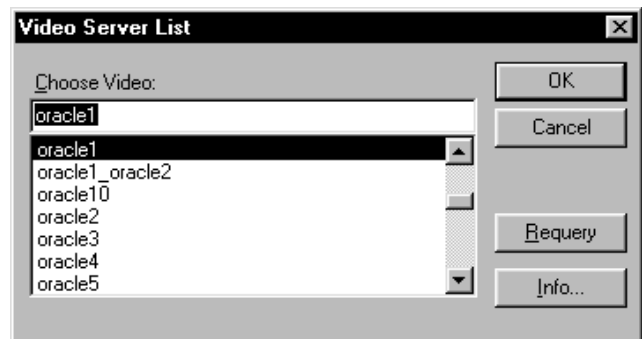


Figure 10 - Dialog Box to Select a Video from Oracle Video Server

Clicking the “Local Video” button opens a preprogrammed, standard file open dialog box that

allows the user to search for and open a video from a local disk. In the code example, see the “Private Sub btnGetLocal\_Click” routine.

Clicking the “Video Info” button opens a form (Figure 11) that is programmed in Visual Basic to show information about the video in progress. The Oracle Video ActiveX Control gathers this information from the Oracle Video Server at runtime.

The screenshot shows a dialog box titled "Video Info" with the following fields:

Stream Name:	/mds/video/oracle1.mpi
Description:	
Format Type:	MPEG 1 Systems
Time Created:	12/17/1997 01:10:38 GMT
Byte Length:	15003136 bytes
Average Bit Rate:	2048000 bps (x 1,000,000)
Total Length:	58606 ms
Encoded Height:	240 pixels
Encoded Width:	352 pixels

An "OK" button is located at the bottom center of the dialog box.

Figure 11 - Form Showing Video File Information

**ActiveX Code for Oracle Video ActiveX Control Sample 2**

The code for this sample is:

```
'
' Oracle Video ActiveX Control Sample 2
'
' - OVC window with ShowControls false
' - Application command buttons for:
'   - get video from video server
'   - get video from local disk
'   - show video file information
'   - play the video
'   - pause the video
'   - resume the video
'   - stop the video
'   - exit the application

'set up GetInfo variables as public, all must be included:
Public streamName As String      'name of the stream
Public url As String             'URL of the stream
Public title As String           'stream description
Public fmtType As String         'format type of the stream
Public extType As String         'readable string for fmtType
Public createTimeStr As String   'time stream was created
Public byteLength As Long        'length of stream in bytes
Public bitrate As Long           'average bit rate
Public presRate As Long          'present bit rate of stream
Public capabilities As Long      'reserved parameter
Public timeLength As Long        'length in milliseconds
Public frameHeight As Integer    'encoded height in pixels
Public frameWidth As Integer     'encoded width in pixels
Public aspectratio As Long       'aspect ratio for the frame
Public frameRate As Long         'current frame rate in fps
Public contentStatus As Integer  'content status
Public protocol As Integer       'network protocol
```

```

'get a video from the video server
Private Sub btnGetServer_Click()
    ovcax1.ImportStreamAs ("")
    ovcax1.Play
End Sub

'get a video from local disk
Private Sub btnGetLocal_Click()
    ovcax1.ImportFileAs
    ovcax1.Play
End Sub

'play the video
Private Sub btnPlay_Click()
    ovcax1.Play
End Sub

'pause the video
Private Sub btnPause_Click()
    ovcax1.Pause
End Sub

'resume video after pause
Private Sub btnResume_Click()
    ovcax1.Resume
End Sub

'rewind the video
Private Sub btnRewind_Click()
    ovcax1.Rewind
End Sub

'stop the video, rewinds to start point
Private Sub btnStop_Click()
    ovcax1.Stop
End Sub

'exit the application
Private Sub btnExit_Click()
    ovcax1.Unload
    Unload Me
End Sub

'get and show the video information parameters
Private Sub btnShowInfo_Click()
    'get the parameters
    Result = ovcax1.GetInfo(streamName, url, title, _
        fmtType, extType, createTimeStr, byteLength, _
        bitrate, presRate, capabilities, timeLength, _
        frameHeight, frameWidth, aspectratio, frameRate, _
        contentStatus, protocol)
    If Result = 1 Then 'if there are parameters, display them
        frmShowInfo.Show 0
    Else 'if no parameters, show message box
        MsgBox "No parameters for video file"
    End If
End Sub

```

```

'
' Form "frmShowInfo" to display selected video file parameters
'
Private Sub Form_Load()
    streamNameTxt.Text = frm2.streamName
    titleTxt.Text = frm2.title
    fmtTypeTxt.Text = frm2.fmtType
    createTimeStrTxt.Text = frm2.createTimeStr
    byteLengthTxt.Text = frm2.byteLength & " bytes"
    bitrateTxt.Text = frm2.bitrate & " bps (x 1,000,000)"
    millisecondsTxt.Text = frm2.timeLength & " ms"
    frameHeightTxt.Text = frm2.frameHeight & " pixels"
    frameWidthTxt.Text = frm2.frameWidth & " pixels"
End Sub

Private Sub OKBtn_Click()
    Unload frmShowInfo
End Sub

```

### Oracle Video ActiveX Control - Sample 3

Figure 12 shows that the Oracle Video ActiveX Control can be placed in a resizable application window. In this example, users can scale the video window both horizontally and vertically by resizing the application window. The Oracle Video ActiveX Control automatically maintains the VCR-style controls for a consistent appearance.

The developer used Visual Basic to resize the application window to fit the ActiveX control. When a user resizes the window at run-time, the application code recognizes this event and resizes the video window to fit within the application window.



Figure 12 - ActiveX Control Sample 3 App with Window Resizing

### ActiveX Code for Oracle Video ActiveX Control Sample 3

The code for this sample is:

```
'
' Oracle Video ActiveX Control Sample 3
'
' - Video window with ShowControls true
' - Application window is sized to fit video window
'   initially at encoded height and width for the video;
'   aspect ratio is determined
' - Resizing of application window at runtime is used to
'   resize video window; aspect ratio is maintained
'
Dim TempVideoHeight As Single 'temporary video height variable
Dim TempVideoWidth As Single 'temporary video width variable
Dim VideoEncodeHeight As Single 'video encoded height in pixels
Dim VideoEncodeWidth As Single 'video encoded width in pixels
Dim VideoAspectRatio As Single 'computed video aspect ratio
Dim VideoControlHeight As Single 'controls height in pixels
Dim HeightConv As Single 'conversion ratio of pixels to twips
Dim WidthConv As Single 'conversion ratio of pixels to twips
Dim HeightAdjust As Single 'height increment of app over video
Dim WidthAdjust As Single 'width increment of app over video
Dim MediaFile As String 'path and name of video file to play

Private Sub Form_Load()

    ' Get conversion ratios for screen pixels to twips.
    HeightConv = Screen.TwipsPerPixelY
    WidthConv = Screen.TwipsPerPixelX

    ' Get window size differences from video window
    ' to app window (frm3).
    HeightAdjust = frm3.Height - ovcax1.Height
    WidthAdjust = frm3.Width - ovcax1.Width

    ' Size video window and application window for encoded
    ' video. An application can provide the encoded height and
    ' width. This is an example for MPEG1 at 240 pixels wide
    ' by 352 pixels high.
    VideoEncodeHeight = 240 * HeightConv
    VideoEncodeWidth = 352 * WidthConv
    '45 pixels for OVCAx1 control height
    VideoControlHeight = 45 * HeightConv

    ' Determine video aspect ratio
    VideoAspectRatio = VideoEncodeHeight / VideoEncodeWidth

    ' Size video window and application window (frm3)
    ovcax1.Height = VideoEncodeHeight + VideoControlHeight
    ovcax1.Width = VideoEncodeWidth
    frm3.Height = ovcax1.Height + HeightAdjust
    frm3.Width = ovcax1.Width + WidthAdjust

    ' Get video file and load it
    MediaFile = "C:\MPEG1content\video7.mpg"
    ovcax1.Load (MediaFile)
End Sub
```

```

Private Sub Form_Resize()

    ' Application window (frm3) has been resized
    ' (i.e. via mouse). Check for minimum video window
    ' height or width with aspect ratio restriction.
    ' Set new video window height and width with aspect ratio
    TempVideoHeight = frm3.Height - VideoControlHeight -
HeightAdjust
    If TempVideoHeight < 0 Then TempVideoHeight = 0
    TempVideoWidth = frm3.Width - WidthAdjust
    If TempVideoWidth < 0 Then TempVideoWidth = 0
    If TempVideoHeight < (TempVideoWidth * VideoAspectRatio)
Then
        TempVideoWidth = (TempVideoHeight / VideoAspectRatio)
    Else
        TempVideoHeight = TempVideoWidth * VideoAspectRatio
    End If

    ' Set height and width for new video window and app window
    ovcax1.Height = TempVideoHeight + VideoControlHeight
    ovcax1.Width = TempVideoWidth
    frm3.Height = ovcax1.Height + HeightAdjust
    frm3.Width = ovcax1.Width + WidthAdjust
End Sub

```

## ORACLE VIDEO JAVA LIBRARY

The Oracle Video Java Library allows developers to use native Java to build applications that work with the Oracle Video Server. The Oracle Video Java Library includes classes and interfaces that set up and control video playback, query the video server for video files, and provide current video information. An extensive set of classes, methods, attributes, and control parameters give the Java developer a great amount of control over Java applications that use video.

### Player Classes

The Player Classes provide the ability to control video file loading and playback and include:

- Player object - controls the appearance of the user interface, playback of video streams, and monitoring of the Player object itself
- PlayerFactory - instantiates the Player object without platform dependence
- PlayerListner - specifies the methods the Player object calls in response to specific events
- PlayerException - handle error and warning situations

The Player object is the central object in the Oracle Java Library and provides the following methods and constants:

Method	Description
addListner()	Register the PlayerListner object (event notification)
getControlComp()	Retrieve Play, Pause, Stop buttons and seek scrollbar
getInfo()	Retrieve information object for current video stream
getPlayer()	Retrieve video screen, controller panel and status

Method	Description
	line
getMinPos()	Get minimum stream position
getPos()	Return current position of video stream
getSelRange()	Retrieve current start and end positions for play-from and play-to
getState()	Return current state of Player object
getStats()	Return object of current playback statistics
getStatusComp()	Retrieve status line component for player interface
getVisualComp()	Retrieve video screen component for player interface
getVol()	Return current volume setting
load()	Load video file specified
pause()	Pause video stream
play()	Play video stream from beginning or play-from position
resume()	Resume play of video stream
setFullScreen()	Set playback in full-screen mode
setPos()	Set video stream to new position
setVol()	Set volume
stateToString()	Convert player state into a text string
stop()	Stop video stream, rewind to beginning or play-from position
term()	Terminate processes for Oracle Video Client
unload()	Unload current video stream, release video server resources

*Table 8 - Oracle Video Java Library Player Methods*

The following constants affect video playback and are contained in the StmOpts class:

Constant	Description
m_autoStart	Specify if video starts automatically when stream is loaded
m_img	Image file spec displayed when video screen is blank
m_leftClick	Specify if left mouse button for play, pause
m_loop	Specify if video is to loop at end of stream
m_playFrom	Specify start point in the video
m_playTo	Specify end point in the video
m_popup	Specify if popup control menu to be displayed
m_volume	Specify volume level

*Table 9 - Oracle Video Java Library Constants*

### Stream Information Classes

The Stream Information Classes check and update the video playback information including stream position, stream information, and stream playback statistics. The current video position can be queried and reset in terms of frame number, milliseconds from the beginning, and hh:mm:ss:cc from the beginning. Video information includes name of the video, transport protocol, file specifier, text description of the video, codec type, transport type, frame height and width, and video creation date. Video playback statistics include current playback state, network protocol, data packets received and dropped, average frame rate, current frame on the video screen, current time position for current frame, and current buffer parameters for the stream data cache.

### Content Query Classes

Content Query Classes query the Oracle Video Server for a list of available videos.

### Oracle Video Java Library - Sample

Figure 13 shows a Java application that uses the Oracle Video Java Library to create a player with on-screen VCR-style controls, a status line, and the popup control menu.



Figure 13 - Java Sample App

### Code for Oracle Video Java Library Sample

The Java code for this sample is:

```
//
// Package oracle.ovc - Oracle Video Client for Java
//
// Simple.java - This example demons the basic Player object
//

// Sun JDK Imports
import java.awt.*;
import java.awt.event.*;

// Oracle Video Client Imports
import oracle.ovc.PlayerFactory;
import oracle.ovc.PlayerException;
import oracle.ovc.Player;

public class Simple
{
    static Player    player        = null;
    static Component playerUI      = null;
    static Frame     playerframe   = null;
    static String    m_MediaFile   = "/mds/video/oracle10.mpi";
    static int       m_Width       = 320;
    static int       m_Height      = 280;

    public static void main (String args[])
    {
        Simple app = new Simple ();
        playerframe = new PlayerFrame ("OVPlayer", m_Width, m_Height);
        app.addPlayer();
        app.loadMediaFile();
    } // end of main

    public void addPlayer()
    {
        // Create a player object.
        try {
            player = PlayerFactory.getPlayer ();
        }
        catch ( UnsatisfiedLinkError e ) {
            System.out.println(e.getMessage());
        }
        catch ( PlayerException e ) {
            System.out.println("Unable to create a Player object.");
        }
        // Create visual playback component of player object.
        try {
            playerUI = player.getPlayerUI ();
        }
        catch ( PlayerException e ) {}

        // Add the visual component of the player object
        // to the frame before invoking init().
        playerframe.add (playerUI);
        playerframe.show ();
    }
}
```

```

        // Invoke init() after the player UI is actually
        // displayed on the window.
        try {
            player.init ();
        }
        catch ( PlayerException e ) {
            System.out.println("init() failed.");
        }
    } // end of addPlayer

    public void loadMediaFile()
    {
        // Load and prepare the default video for playback
        // with default stream options.
        try {
            player.load (m_MediaFile, null);
        }
        catch ( PlayerException e ) {
            System.out.println("Could not load content.");
        }
    } // end of initial_load
} // end of Simple class

class PlayerFrame extends Frame
{
    public PlayerFrame (String title, int m_Width, int m_Height)
    {
        super (title);
        // Register a Windowlistener after creating the frame,
        // the listener is required to exit properly from app.
        addWindowListener(new PFAdapter());
        setSize (m_Width, m_Height);
    } // end of constructor
} // end of class

class PFAdapter extends WindowAdapter
{
    public void windowClosing(WindowEvent event)
    {
        System.out.println("Shutting down the java client ...");
        if (Simple.player != null)
        {
            try {
                System.out.println("Player state before exiting: " +
Simple.player.getState());
                // Unload the video before exiting, thereby releasing
                // any resources associated with the current stream.
                Simple.player.unload ();
                Simple.player.term ();
            }
            catch (PlayerException e) {}
        }
        System.exit(0);
    } // end of windowClosing
} // end of PFAdapter

```

## ORACLE VIDEO SERVER APPLICATION PROGRAMMING INTERFACES

In addition to the tools described in the previous sections for developing client applications for the Oracle Video Server, system developers can use the public APIs to build highly customized systems. The Oracle Video Server has a modular architecture that adheres to industry standards.

For example, the Oracle Video Server APIs can be used for the following tasks:

- Integrating varied client devices such as set-top boxes, network computers, and kiosks
- Developing utilities for managing stored video content and its usage, such as for media asset management
- Integrating real-time encoders that feed Oracle Video Server using the public Video Encoding Standard (VES) specifications
- Integrating hierarchical storage systems
- Developing utilities for scheduling and managing video-on-demand
- Implementing specialized security controls
- Monitoring system activity for extended usage statistics and billing purposes

The Oracle Video Server system uses various services for specific functions. These services are:

- **Session Service** - establishes and maintains the client/server communication channels and manages a set of Oracle Video Server resources on behalf of a particular client device.
- **Content Service** - Provides the client with information on the content available to it from the Oracle Video Server system.
- **Stream Service** - instructs the “video pump” on the Oracle Video Server to stream video to the client and performs rate control operations.
- **Media Data Store Service** - stores content on the Oracle Video Server and provides access to all content in the system.

Each of these services can be manipulated through an API. The Oracle Video Server APIs adhere to the industry-standard Network Computing Architecture. This architecture supports an Object Request Broker (ORB) via the CORBA Internet Inter-Orb Protocol (IIOP) backbone and supports middleware protocols such as HTTP and SQL\*Net. The Oracle Video Server APIs are provided in the standard Interface Definition Language (IDL) format.

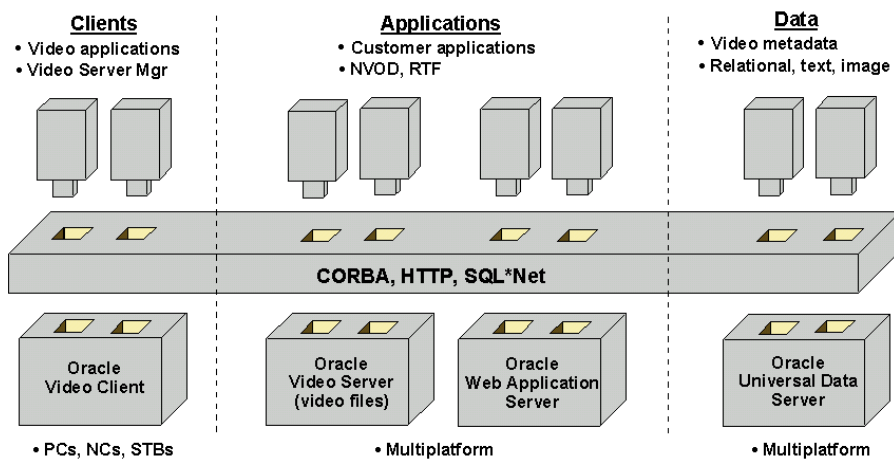


Figure 14 - Oracle Video Server and the Network Computing Architecture

For more information about the Oracle Video Server APIs, see the *Oracle Video Server Developer's Guide*.

Adding Video to Applications with the Oracle Video Server  
March 1998  
Author: Martin McKendrick  
Copyright © Oracle Corporation 1998  
All Rights Reserved Printed in the U.S.A.

This document is provided for informational purposes only and the information herein is subject to change without notice. Please report any errors herein to Oracle Corporation. Oracle Corporation does not provide any warranties covering and specifically disclaims any liability in connection with this document.

Oracle is a registered trademark and Enabling the Information Age, Oracle Video Server, and Network Computers are trademarks of Oracle Corporation.

All other company and product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

**ORACLE**

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
415.506.7000  
Fax 415.506.7200  
<http://www.oracle.com/>

Copyright © Oracle Corporation 1998  
All Rights Reserved