



NETWORK CAMERA Protocol Spec. HTTP Protocol Tutorial

Ver. 1.0

April 16, 2009
CANON INC.

Change Tracking List

Version	Date	Revised page	Note
Ver. 1.0	April 16, 2009		First Edition

Table of Contents

1	Overview	1
2	Displaying Images.....	2
2.1	Displaying Still Images on Web Browser.....	2
2.2	Displaying Still Images using Java Program	2
2.3	Displaying Video using Java Program.....	4
3	Controlling Camera	6
3.1	Controlling Camera using Web browser.....	6
3.2	Controlling Camera using Java Program.....	7
4	Application with Camera Control Privilege.....	13
5	Example using C#	23
5.1	Creating GUI	23
5.2	Creating Session Class	26
5.3	Combining GUI and Session	29
6	Conclusion.....	36

1 Overview

This tutorial is designed

- for those who want to learn WebView protocol (WV-HTTP) in HTTP version
- to give them a better understanding of the outline of WV-HTTP through creating samples.

To use this tutorial, you are required to have previous knowledge of both

- basic knowledge of HTTP, and
- basic programming technique.

Although examples in this tutorial are all written in Java and C# languages, it is easy to understand for those who are conversant with either language since both languages are similar in many ways. The performance check of the samples was conducted with JDK1.6.0 and Visual C# 2005/2008 Express Edition.

In this tutorial, the sample program commands are explained using VB-C60 as an example. However, you can also use this tutorial for VB-C500 series by simply replacing the model name 'VB-C60' with 'VB-C500'. However, commands you can use depend on the camera model.

2 Displaying Images

This section explains how to display images from a WebView camera. In this tutorial, "172.20.28.60" is used as a camera server's IP address, but it is a virtual address used for convenience of explanation. So, you need to replace the address with your IP address.

2.1 Displaying Still Images on Web Browser

Start Internet Explorer, Firefox or other web browser and enter the following URL, so that the current image (still image) will be displayed.

<http://172.20.28.60/-wvhttp-01-/GetOneShot>

2.2 Displaying Still Images using Java Program

Hereafter, we explain how to create network camera applications step by step. First, start with an application that is to obtain and display one still image.

SimpleStillImage.java
<pre>import java.awt.*; import java.awt.image.*; import java.awt.event.*; import java.net.*; import javax.imageio.*; /** * Window class * * Pop up a window and draw an image on it. */ class WvFrame extends Frame { private Image liveImage; WvFrame() { super("Network Camera Sample Application"); setSize(320, 260); addWindowListener(new WindowAdapter() { public void windowClosing(WindowEvent windowEvent) { System.exit(0); } }); } public void setImage(Image image) { liveImage = image; repaint(); } }</pre>

```

    }

    public void paint(Graphics g) {
        if(liveImage != null)
            g.drawImage(liveImage, 0, 20, null);
    }

    public void update(Graphics g) {
        paint(g);
    }
}

/**
 * Main class
 *
 * Create a window and display live images obtained from a camera server on the window.
 */
public class SimpleStillImage {

    public static void main(String[] args) throws Exception {

        // 0) Create and display a window.
        WvFrame f = new WvFrame();
        f.setVisible(true);

        // 1) Create the URL for obtaining images.
        URL url = new URL("http://172.20.28.60/-wvhttp-01-/GetOneShot");

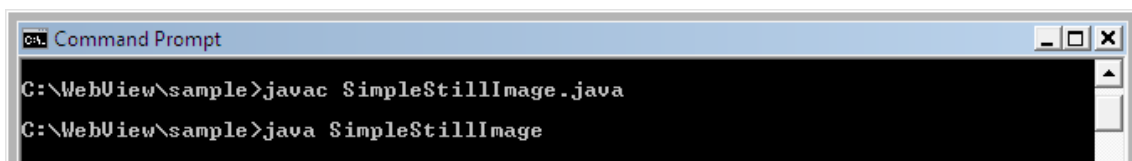
        // 2) Connect to a server.
        URLConnection con = url.openConnection();

        // 3) Create an Image object from a server's response.
        BufferedImage image = ImageIO.read(con.getInputStream());

        // 4) Display obtained images on the window.
        f.setImage(image);
    }
}

```

Save this file to "SimpleStillImage.java", and then compile and execute it as shown below.



```

C:\WebVew\sample>javac SimpleStillImage.java
C:\WebVew\sample>java SimpleStillImage

```

As the execution result, one still image will be displayed as shown in the following image.



2.3 Displaying Video using Java Program

You can display motion video by obtaining and displaying still images periodically. To create video application, substitute the “main” function with the code below from earlier-explained “SimpleStillImage.java”. Note that you need to rename the name of the main class too.

SimpleLiveImage.java
<pre> /** * Main class * * Create a window and display live images obtained from a camera server on the window. */ public class SimpleLiveImage { public static void main(String[] args) throws Exception { // 0) Create the URL for obtaining images. WvFrame f = new WvFrame(); f.setVisible(true); while(true) { // 1) Create the URL for obtaining images. URL url = new URL("http://172.20.28.60/~wvhttp-01-/GetOneShot"); // 2) Connect to a server. URLConnection con = url.openConnection(); // 3) Create an Image object from a server's response. BufferedImage image = ImageIO.read(con.getInputStream()); // 4) Display obtained images on the window. </pre>

```
f.setImage(image);  
  
    // 5) Sleep.  
    Thread.sleep(500);  
}  
}  
}
```

You can adjust the drawing rate by changing an interval of Sleep.

3 Controlling Camera

3.1 Controlling Camera using Web browser

Start InternetExplorer, Firefox or other web browser, and then enter the following URL.

<http://172.20.28.60/-wvhttp-01-/control?pan=100&tilt=-10&zoom=50>

The camera will move, and its pan, tilt, zoom values will be displayed in the browser as follows.

```
c.1.zoom:=50
c.1.pan:=100
c.1.tilt:=-10
```

Input various values for pan, tilt and zoom to check if the camera moves properly in response to the input values.

To check the range that can be applied to pan, tilt and zoom, enter the following URL.

<http://172.20.28.60/-wvhttp-01-/GetCameraInfo>

The camera will move and following values will be displayed.

```
camera_type=Canon VB-C60
camera_id=1
camera_status=enabled
pan_current_value=100
tilt_current_value=-10
zoom_current_value=50
back_light=OFF
pan_left_end=-17000
pan_right_end=17000
tilt_up_end=9000
tilt_down_end=-2500
zoom_tele_end=39
zoom_wide_end=5580
pan_left_limit=-17000
pan_right_limit=17000
tilt_up_limit=9000
tilt_down_limit=-2500
zoom_tele_limit=39
zoom_wide_limit=5580
view_left_boundary=-19790
view_right_boundary=19790
view_up_boundary=11093
view_down_boundary=-4593
view_tele_boundary=39
view_wide_boundary=5580
focus_mode=auto
white_balance=auto
```

“pan_current_value” shows the current pan value, “pan_left_end” shows the minimum pan value, and “pan_right_end” shows the maximum pan value. The same holds for tilt and

zoom. "..._end" and "..._limit" for pan, tilt and zoom have a following meaning respectively.

- ...end = the limit value of camera's movement
- ...limit = the limit value that can be specified with a parameter

Normally, those two values show the same value. Camera servers, however, return different values when view restrictions are set. Please refer to the manual for other parameters.

To check if the camera was actually controlled, access to the following URL and obtain images.

<http://172.20.28.45/-wvhttp-01-/GetOneShot>

3.2 Controlling Camera using Java Program

So far, you have learned a way to display motion video, a command to obtain the camera information, and a command to control a camera. This section explains how to create an application for controlling a camera with using a combination of those commands.

SimpleCtrlViewer.java
<pre>import java.awt.*; import java.awt.image.*; import java.awt.event.*; import java.io.*; import java.net.*; import javax.imageio.*; /** * Window class * * Create and lay out a scrollbar for PTZ. */ class WvFrame extends Frame implements AdjustmentListener { private WvCanvas canvas; private Scrollbar pBar, tBar, zBar; private Button button; WvFrame() throws Exception { super("Network Camera Control Application"); setSize(400, 300); // Scrollbar button tBar = new Scrollbar(Scrollbar.VERTICAL); pBar = new Scrollbar(Scrollbar.HORIZONTAL);</pre>

```

zBar = new Scrollbar(Scrollbar.VERTICAL);

pBar.addAdjustmentListener(this);
tBar.addAdjustmentListener(this);
zBar.addAdjustmentListener(this);

// Image display area
canvas = new WvCanvas();

// Layout each part
Panel p0 = new Panel();
p0.setLayout(new GridLayout(1,2,10,0));
p0.add(tBar);
p0.add(zBar);

setLayout(new BorderLayout(10,10));
add("Center", canvas);
add("East", p0);
add("South", pBar);

// Obtain the camera information (PTZ range etc) and initialize the GUI.
String str = sendCommand("GetCameraInfo");
processServerResponse(str);

addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
});
}

/**
 * Interpret a response from a camera server and renew the GUI.
 */
public void processServerResponse(String response) throws Exception
{
    int pVal = Integer.MIN_VALUE, tVal = Integer.MIN_VALUE, zVal = Integer.MIN_VALUE;

    BufferedReader br = new BufferedReader(new StringReader(response));
    while(true) {
        String line = br.readLine();
        if(line == null) {
            break;
        }
        else if(line.startsWith("pan_current_value=")||line.startsWith("pan="))
        {
            pVal = getValue(line);
        }
        else if(line.startsWith("tilt_current_value=")||line.startsWith("tilt="))
        {
            tVal = getValue(line);
        }
        else if(line.startsWith("zoom_current_value=")||line.startsWith("zoom="))

```

```

    {
        zVal = getValue(line);
    }
    else if(line.startsWith("pan_left_end=")) {
        pBar.setMinimum(getValue(line));
    }
    else if(line.startsWith("pan_right_end=")) {
        pBar.setMaximum(getValue(line));
    }
    else if(line.startsWith("tilt_up_end=")) {
        tBar.setMaximum(getValue(line));
    }
    else if(line.startsWith("tilt_down_end=")) {
        tBar.setMinimum(getValue(line));
    }
    else if(line.startsWith("zoom_tele_end=")) {
        zBar.setMinimum(getValue(line));
    }
    else if(line.startsWith("zoom_wide_end=")) {
        zBar.setMaximum(getValue(line));
    }
}

// Note: Please note that the tilt range is upside down.
if(pVal != Integer.MIN_VALUE)
    pBar.setValue(pVal);
if(tVal != Integer.MIN_VALUE)
    tBar.setValue( tBar.getMaximum() + tBar.getMinimum() - tVal );
if(zVal != Integer.MIN_VALUE)
    zBar.setValue(zVal);
}

private int getValue(String line)
{
    int p = line.indexOf('=');
    if( p > 0 )
        return Integer.parseInt(line.substring(p+1));
    return 0;
}

/**
 * Display an image.
 */
public void setImage(Image image) {
    canvas.liveImage = image;
    canvas.repaint();
}

/**
 * Callback function called when the scrollbar is controlled.
 */

```

```

public void adjustmentValueChanged(AdjustmentEvent e){
    if(e.getValueIsAdjusting())
        return;

    Object src = e.getSource();
    int val = e.getValue();
    if(src == pBar) {
        sendCommand("control?pan="+val);
    }
    else if(src == tBar) {
        sendCommand("control?tilt="+tBar.getMaximum() + tBar.getMinimum() - val));
    }
    else if(src == zBar) {
        sendCommand("control?zoom="+val);
    }
}

/**
 * Send a command to a camera server and return the response.
 */
private String sendCommand(String command)
{
    try {
        URL url = new URL("http://172.20.28.60/-wvhttp-01-/" + command);
        URLConnection con = url.openConnection();
        InputStreamReader isr = new InputStreamReader(con.getInputStream());
        BufferedReader br = new BufferedReader(isr);
        StringBuilder sb = new StringBuilder();
        while (true) {
            String line = br.readLine();
            if (line == null)
                break;
            sb.append(line+"¥n");
        }
        return sb.toString();
    } catch (Exception e) {}
    return null;
}

/**
 * Internal class for displaying live images.
 */
class WvCanvas extends Canvas {
    private Image livelImage;

    public void paint(Graphics g) {
        if (livelImage != null) {
            Dimension d = getSize();
            g.drawImage(livelImage, 0, 0, d.width, d.height, null);
        }
    }
}

```

```

    public void update(Graphics g) {
        paint(g);
    }

    public Dimension getPreferredSize() {
        return new Dimension(320, 240);
    }
}

/**
 * Main class
 *
 * Create a window and display live images obtained from a camera server on the window.
 */
public class SimpleCtrlViewer {

    public static void main(String[] args) throws Exception {

        // 0) Create and display a window.
        WvFrame f = new WvFrame();
        f.setVisible(true);

        while(true)
        {
            // 1) Create the URL for obtaining images.
            URL url = new URL("http://172.20.28.60/-wvhttp-01-/GetOneShot");

            // 2) Connect to a server.
            URLConnection con = url.openConnection();

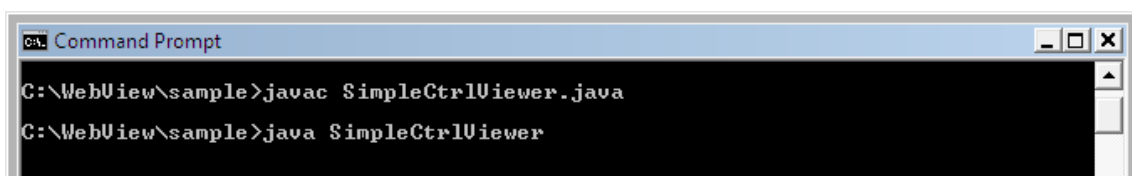
            // 3) Create an Image object from a server's response.
            BufferedImage image = ImageIO.read(con.getInputStream());

            // 4) Display obtained images on the window.
            f.setImage(image);

            // 5) Sleep.
            Thread.sleep(500);
        }
    }
}

```

After saving this file to " SimpleCtrlViewer.java", compile and execute it as follows.

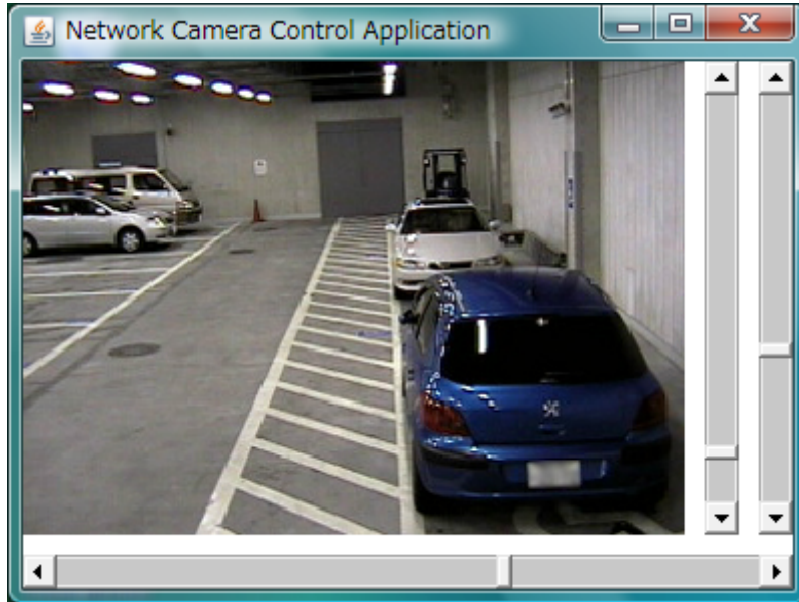


```

C:\WebView\sample>javac SimpleCtrlViewer.java
C:\WebView\sample>java SimpleCtrlViewer

```

As the execution result, viewer application, which enables you to control pan, tilt and zoom, will start up, as shown in the following image.



4 Application with Camera Control Privilege

The application “SimpleCtrlViewer”, which has already explained in the previous section, works without a problem while a single user is connecting to a camera. However, it might not work properly if multiple users connect to one camera at the same time. This is because, with the application “SimpleCtrlViewer”, the very last operation is applied. Thus, some users’ operations might not be reflected as they expect.

To avoid such situation as multiple users control one camera at the same time, WV-HTTP protocols have adopted a concept of “camera control privilege”, which allows only one user with the privilege to control the camera.

The procedure is as follows.

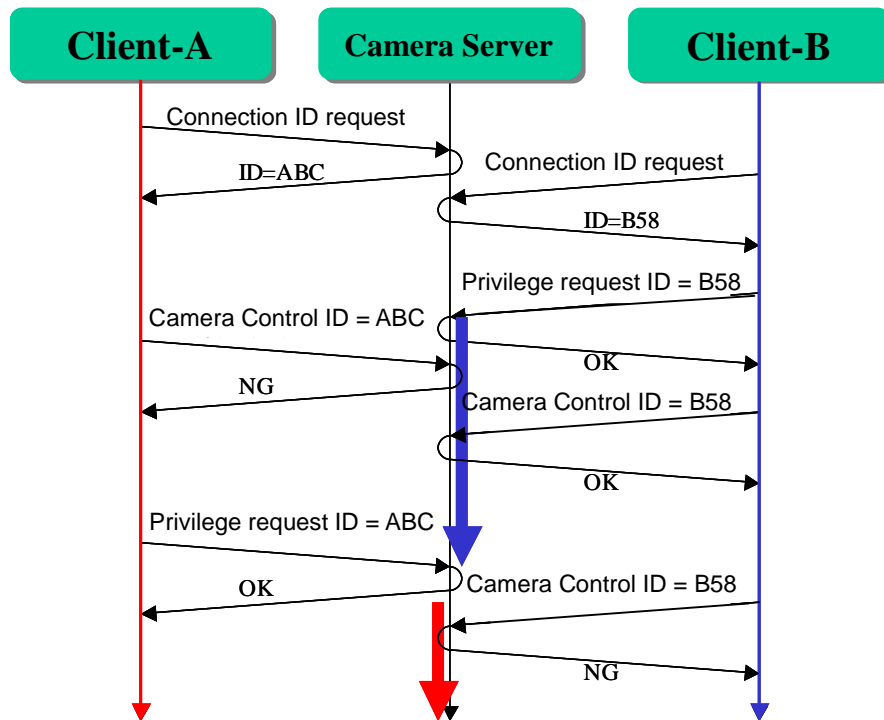
- 1) First, users who want to control a camera need to obtain the camera control privilege.
- 2) Users who obtained the privilege can control the camera for a given period of time.
- 3) If users couldn’t obtain the privilege because another user has already obtained it, the users need to wait in a queue for the privilege until it is his/her turn.

WV-HTTP is based on HTTP protocols. In HTTP protocols, each transaction is independent of each other, which is called state-less. However, when each transaction is independent, who is holding the privilege is unknown.

So, WV-HTTP manages sessions through the following procedure.

- A camera server assigns a unique text called connection ID to each client, when clients first connect to the camera server.
- Clients add own connection ID to their requests to the camera server.
- The server identifies whose request it is by checking the connection ID.

Following figure shows client A and B are alternately obtaining a camera control privilege using own connection IDs.



Following is an application with a PTZ control privilege.

CameraCtrlViewer.java
<pre> import java.awt.*; import java.awt.event.*; import java.io.*; import java.net.*; import java.util.regex.*; import javax.imageio.*; /** * Window class * * Create and lay out a scrollbar for PTZ and a camera control button. */ class WvFrame extends Frame implements AdjustmentListener, ActionListener { private WvCanvas canvas; private Scrollbar pBar, tBar, zBar; private Button button; private WvSession session; </pre>

```

WvFrame(WvSession session) throws Exception
{
    super("Network Camera Control Application");
    setSize(400, 300);

    this.session = session;

    // Camera control button
    button = new Button(" push ");
    button.addActionListener(this);
    button.setBackground(Color.yellow);

    // Scrollbar button
    tBar = new Scrollbar(Scrollbar.VERTICAL);
    pBar = new Scrollbar(Scrollbar.HORIZONTAL);
    zBar = new Scrollbar(Scrollbar.VERTICAL);

    pBar.addAdjustmentListener(this);
    tBar.addAdjustmentListener(this);
    zBar.addAdjustmentListener(this);

    // Image display area
    canvas = new WvCanvas();

    // Layout each part
    Panel p0 = new Panel();
    p0.setLayout(new GridLayout(1,2,10,0));
    p0.add(tBar);
    p0.add(zBar);

    Panel p1 = new Panel();
    p1.setLayout(new BorderLayout());
    p1.add("Center", pBar);
    p1.add("East", button);

    setLayout(new BorderLayout(10,10));
    add("Center", canvas);
    add("East", p0);
    add("South", p1);

    // Obtain the camera information (PTZ range etc) and initialize the GUI.
    String s t r = sendCommand("GetCameraInfo");
    processServerResponse(s t r);

    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
}

```

```

/**
 * Interpret an event notification from a camera server and renew the GUI.
 */
public void processServerResponse(String response) throws Exception
{
    int pVal = Integer.MIN_VALUE, tVal = Integer.MIN_VALUE, zVal = Integer.MIN_VALUE;

    BufferedReader br = new BufferedReader(new StringReader(response));
    while(true) {
        String line = br.readLine();
        if(line == null) {
            break;
        }
        else if((line.startsWith("pan_current_value=")||line.startsWith("pan="))
        {
            pVal = getValue(line);
        }
        else if((line.startsWith("tilt_current_value=")||line.startsWith("tilt="))
        {
            tVal = getValue(line);
        }
        else if((line.startsWith("zoom_current_value=")||line.startsWith("zoom="))
        {
            zVal = getValue(line);
        }
        else if((line.startsWith("pan_left_limit=")) {
            pBar.setMinimum(getValue(line));
        }
        else if((line.startsWith("pan_right_limit=")) {
            pBar.setMaximum(getValue(line));
        }
        else if((line.startsWith("tilt_up_limit=")) {
            tBar.setMaximum(getValue(line));
        }
        else if((line.startsWith("tilt_down_limit=")) {
            tBar.setMinimum(getValue(line));
        }
        else if((line.startsWith("zoom_tele_end=")) {
            zBar.setMinimum(getValue(line));
        }
        else if((line.startsWith("zoom_wide_end=")) {
            zBar.setMaximum(getValue(line));
        }
        else if((line.startsWith("enabled_camera_control"))
        {
            button.setLabel("control");
            button.setBackground(Color.green);
        }
        else if((line.startsWith("disabled_camera_control") ||
            line.startsWith("failed_to_get_camera_control"))
        {
            button.setLabel(" push ");
            button.setBackground(Color.yellow);
        }
    }
}

```

```

        else if(line.startsWith("waiting_camera_control"))
        {
            button.setLabel("waiting");
            button.setBackground(Color.magenta);
        }
    }

    // Note: Please note that the tilt range is upside down.
    if(pVal != Integer.MIN_VALUE)
        pBar.setValue(pVal);
    if(tVal != Integer.MIN_VALUE)
        tBar.setValue( tBar.getMaximum() + tBar.getMinimum() - tVal );
    if(zVal != Integer.MIN_VALUE)
        zBar.setValue(zVal);
}

private int getValue(String line)
{
    int p = line.indexOf('=');
    if( p > 0 )
        return Integer.parseInt(line.substring(p+1));
    return 0;
}

/**
 * Display an image.
 */
public void setImage(Image image) {
    canvas.liveImage = image;
    canvas.repaint();
}

/**
 * Send a command to a camera server and return the response. (Actual process is done by the
session object.) */
private String sendCommand(String command)
{
    try {
        return session.sendCommand(command);
    } catch (Exception e) {
        return null;
    }
}

/**
 * Callback function called when the control button is pressed
 */
public void actionPerformed(ActionEvent e) {
    sendCommand("GetCameraControl");
}

```

```

/**
 * Callback function called when the scroll bar is controlled
 */
public void adjustmentValueChanged(AdjustmentEvent e){
    if(e.getValueIsAdjusting())
        return;

    Object src = e.getSource();
    int val = e.getValue();
    if(src == pBar) {
        sendCommand("control?pan="+val);
    }
    else if(src == tBar) {
        sendCommand("control?tilt="+tBar.getMaximum() + tBar.getMinimum() - val);
    }
    else if(src == zBar) {
        sendCommand("control?zoom="+val);
    }
}

/**
 * Internal class for displaying live images
 */
class WvCanvas extends Canvas {
    private Image liveImage;

    public void paint(Graphics g) {
        if (liveImage != null) {
            Dimension d = getSize();
            g.drawImage(liveImage, 0, 0, d.width, d.height, null);
        }
    }

    public void update(Graphics g) {
        paint(g);
    }

    public Dimension getPreferredSize() {
        return new Dimension(320, 240);
    }
}

/**
 * Class for managing a session with a camera server
 *
 * Create a thread and execute the GetNotice command to a camera server.
 */
class WvSession implements Runnable {
    String server;

```

```

String connectionId;
Thread thread;
WvFrame frame;

/**
 * Connect to a camera server to obtain a session ID.
 */
public WvSession(String server) throws Exception {
    this.server = server;
    String str = sendCommand("OpenCameraServer");

    // Extract connection_id from the response.
    Pattern pattern = Pattern.compile("connection_id=(.+)");
    Matcher matcher = pattern.matcher(str);
    if(matcher.find())
        connectionId = matcher.group(1);

    thread = new Thread(this);
    thread.start();
}

public void addListener(WvFrame frame) {
    this.frame = frame;
}

/**
 * Main thread of session class
 * Repeat the GetNotice command and notify the result to the window.
 */
public void run() {
    try {
        while(thread != null) {
            String str = sendCommand("GetNotice");
            if (frame != null) {
                frame.processServerResponse(str);
            }
            Thread.sleep(200);
        }
    } catch (Exception e) {}
}

/**
 * Obtain images from a camera server.
 */
public Image getImage() throws Exception {
    URL url = new URL(server + "GetLiveImage?connection_id=" + connectionId);
    URLConnection con = url.openConnection();
    return ImageIO.read(con.getInputStream());
}

```

```

/**
 * Send a command to a camera server and return the response.
 */
public String sendCommand(String command) throws Exception {
    if(connectionId != null) {
        if (command.indexOf("?") > 0) {
            command = command + "&connection_id=" + connectionId;
        } else {
            command = command + "?connection_id=" + connectionId;
        }
    }

    URL url = new URL(server + command);
    URLConnection con = url.openConnection();
    InputStreamReader isr = new InputStreamReader(con.getInputStream());
    BufferedReader br = new BufferedReader(isr);
    StringBuilder sb = new StringBuilder();
    while (true) {
        String line = br.readLine();
        if (line == null)
            break;
        sb.append(line+"\n");
    }
    return sb.toString();
}

/**
 * Main class
 *
 * Create a window and a session, and then associate them with each other.
 * Also, display a live image obtained from a camera server on the window.
 */
public class CameraCtrlViewer {

    public static void main(String[] args) throws Exception {

        // 0) Create a session.
        WvSession session = new WvSession("http://172.20.28.60/-wvhttp-01-");

        // 1) Create and display a window.
        WvFrame f = new WvFrame(session);
        f.setVisible(true);

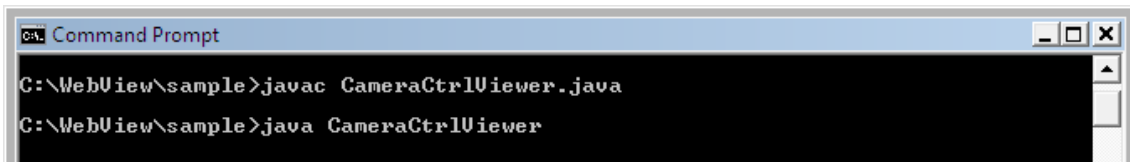
        // 2) Pass the window to the session object.
        session.addListener(f);

        // 3) Renew the image.
        while(true) {
            f.setImage(session.getImage());
            Thread.sleep(500);
        }
    }
}

```

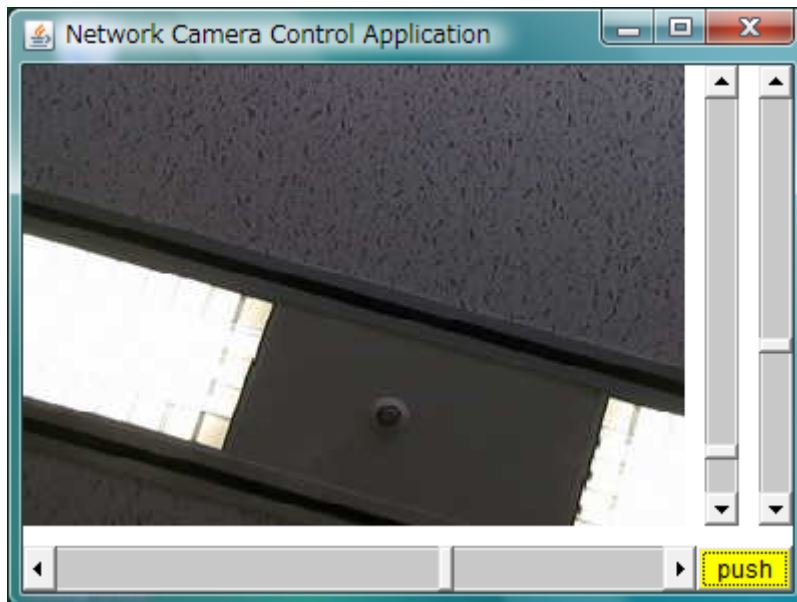
```
}  
}
```

Save this file to the file "CameraCtrlViewer.java", and then compile and execute it as shown below.



```
C:\WebView\sample>javac CameraCtrlViewer.java  
C:\WebView\sample>java CameraCtrlViewer
```

As the execution result, a window with camera control privilege will be displayed as shown in the following image.



Click the "push" button to send a request for the camera control privilege. If you obtain the privilege, the button will turn green and you can control the camera.

To check how it works, start up two or more of this application at the same time and check

- if the camera control privilege switches from one client to another by pressing the "push" button, and
- if one client's operation reflects properly to another window (status of the scrollbar)

Please note that in this example, a thread is created in the WvSession class. In WV-HTTP protocols, a GetNotice command is used to detect status changes on a camera server, such as the control privilege and PTZ values etc. You need to pay attention that GetNotice requests cannot be executed from the main thread or a GUI thread because GetNotice

requests remain blocked until any change occurs. This is because events on camera servers occur asynchronously. For those reasons, in WvSession class, a GetNotice-specific thread is created.

Pan/Tilt/Zoom scrollbars have been added to WvFrame. Please note that these scrollbars provide following two functions.

- Controlling a camera
- Notifying users of the current status of the camera

In Java, the top position of the vertical scrollbar represents the minimum value and the bottom position represents the maximum value. On the other hand, in camera's tilt control, the top position means the maximum value and the bottom position means the minimum value, as shown in the following figure.

	Vertical scrollbar in Java	Camera's tilt control
Top position	Minimum value	Maximum value
Bottom position	Maximum value	Minimum value

It means, for example, when you want to move the camera upward, you need to move the scrollbar downward, which is an opposite direction of the camera movement.

That's why, in WvFrame, the tilt scrollbar is programmed in the opposite way for intuitive control, so that cameras can be controlled in the same direction as the scrollbar.

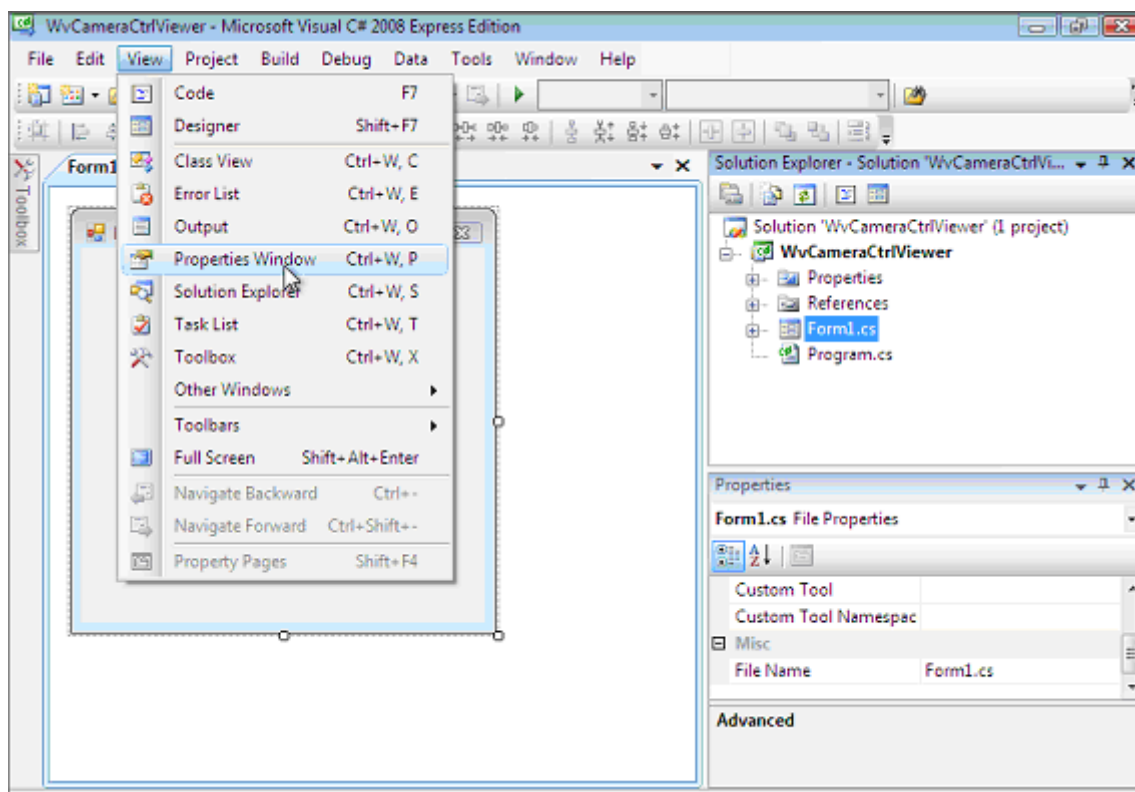
5 Example using C#

So far, you have learned examples using the Java language. This section describes examples using the C# language, but the basic structure is the same as the Java language.

5.1 Creating GUI

From the “File” menu, choose “New Project” > “Windows application”. Enter “WvCameraCtrlViewer” as the project name and press OK.

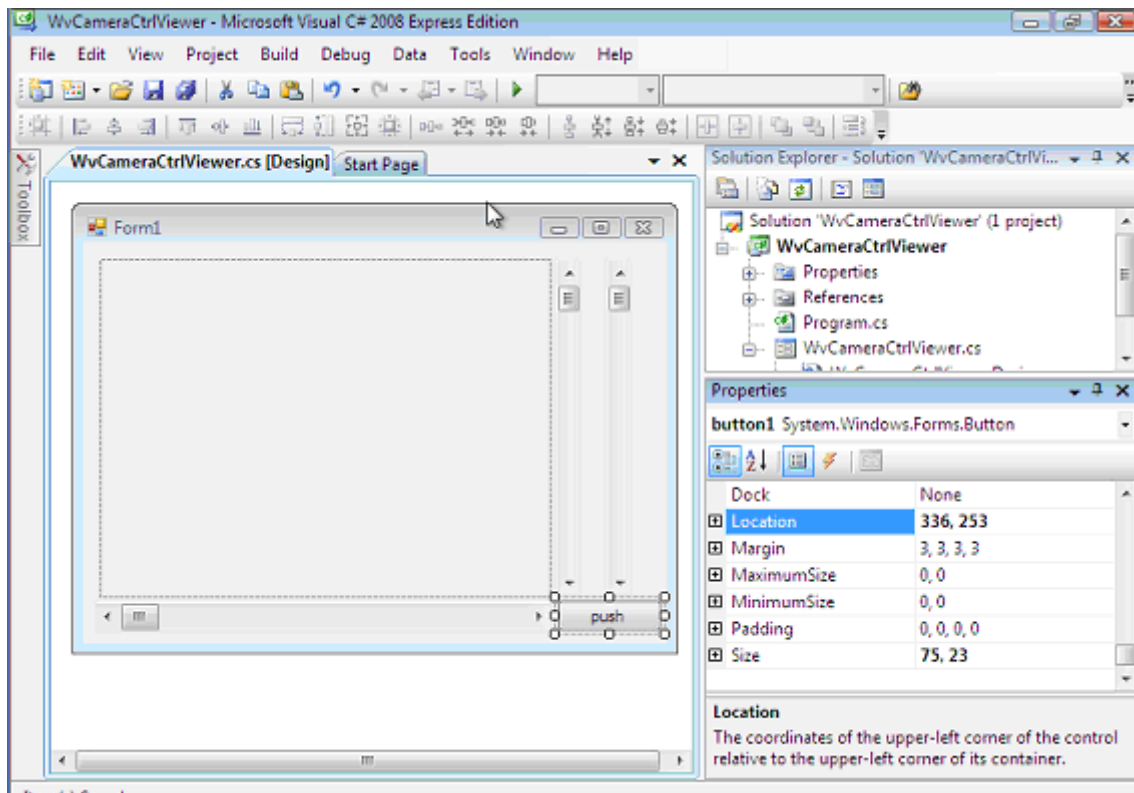
Choose “Form1.cs” in Solution Explorer. Then, choose “View” > “Property Window” and change the file name to “WvCameraCtrlViewer”.



Click “Yes” if a dialog box saying “You are renaming a file. Would you also like to perform a rename in this project of all references to the code element ‘Form1’?” appears.

Choose Form1 in Design and set the size to “430,320” in the property window. From the View menu, choose “Tool Box” to display the toolbox. Then, drag and drop following items onto the Form1.

	Part name	Property name	Property value
Live video screen	pictureBox	(Name)	liveScreen
		Size	320,240
		Location	12,12
		SizeMode	StretchImage
Pan scrollbar	HScrollBar	(Name)	pScrollBar
		Size	323,21
		Location	9,255
Tilt scrollbar	VScrollBar	(Name)	tScrollBar
		Size	17,237
		Location	336,13
Zoom scrollbar	VScrollBar	(Name)	zScrollBar
		Size	17,237
		Location	372,13
Camera control button	Button	(Name)	controlButton
		Size	75,23
		Location	336,253
		Text	push



Next, register an event handler to pan, tilt, and zoom scrollbars and a button.

After choosing a component on the Form, choose an event (⚡) in the property window and then double click the blank column, next to the event name. Following is the events and functions to be registered.

	Component name	Event name	Function name
Camera control button	controlButton	Click	controlButton_Click
Pan scrollbar	pScrollBar	Scroll	pScrollBar_Scroll
Tilt scrollbar	tScrollBar	Scroll	tScrollBar_Scroll
Zoom scrollbar	zScrollBar	Scroll	zScrollBar_Scroll
Main form	WvCameraCtrlViewer	Load	WvCameraCtrlViewer_Load
	WvCameraCtrlViewer	FormClosing	WvCameraCtrlViewer_FormClosing

Check if following functions have been added to “WvCameraCtrlViewer.cs”.

```
private void controlButton_Click(object sender, EventArgs e)
{
}

private void pScrollBar_Scroll(object sender, ScrollEventArgs e)
{
}

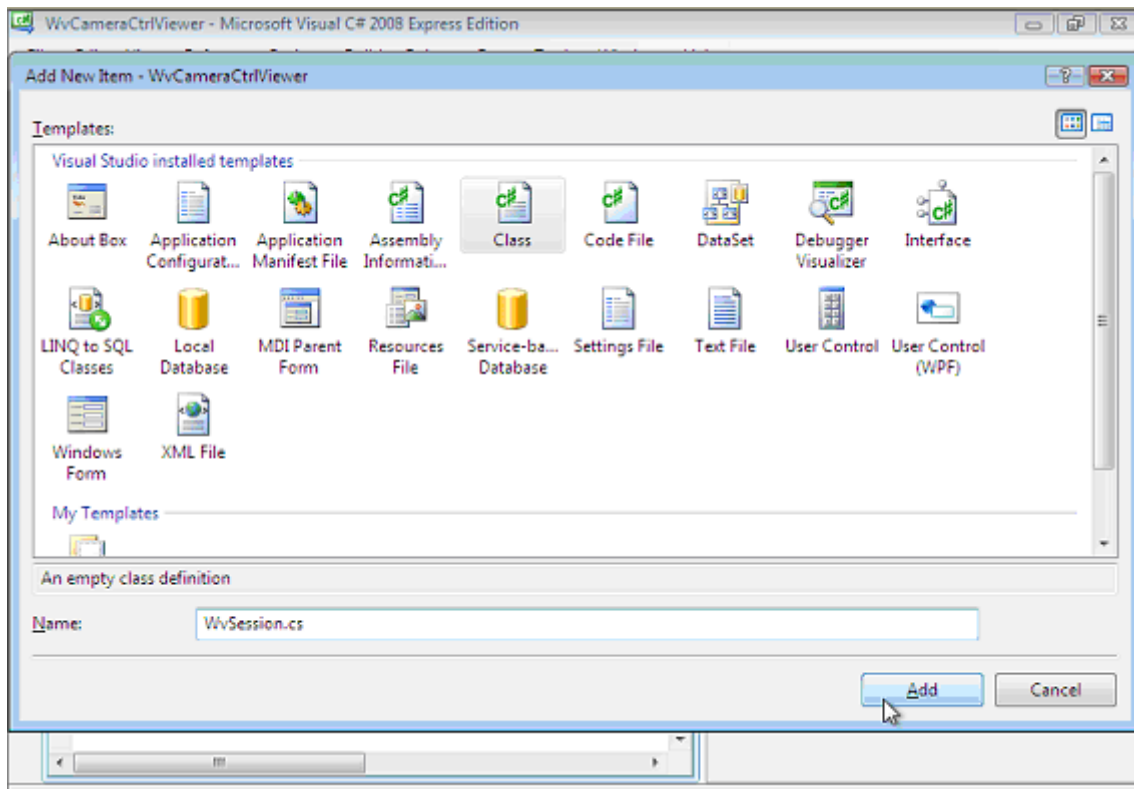
private void tScrollBar_Scroll(object sender, ScrollEventArgs e)
```

```
{  
}  
  
private void zScrollBar_Scroll(object sender, ScrollEventArgs e)  
{  
}  
  
private void WvCameraCtrlViewer_Load(object sender, EventArgs e)  
{  
}  
  
private void WvCameraCtrlViewer_FormClosing(object sender, FormClosingEventArgs e)  
{  
}
```

From the File menu, choose “Save All” to save all of the files. After that, choose “Debug” > “Start Debugging” to execute and check if the compiling and executing are carried out properly.

5.2 Creating Session Class

Press Ctrl-Shift-A to open the “Add New Item” dialog box. Choose a class and enter “WvSession.cs” for the file name, and then press “Add”.



Following is contents of the file.

WvSession.cs
<pre> using System; using System.Collections.Generic; using System.Text; using System.Text.RegularExpressions; using System.IO; using System.Net; using System.Drawing; using System.Threading; namespace WvCameraCtrlViewer { delegate void WvEventCallback(string str); /// <summary> /// Class for managing a session /// </summary> class WvSession : IDisposable { private string server; private string connectionId; private Thread thread; public event WvEventCallback OnGetNotice; </pre>

```

public WvSession(string server)
{
    this.server = server;
    string str = SendCommand("OpenCameraServer");

    // Extract connection_id from the response.
    Match m = Regex.Match(str, "connection_id=(.+)");
    if(m.Success)
        connectionId = m.Groups[1].Value;

    // Start a thread.
    thread = new Thread(new ThreadStart(GetNotice));
    thread.Start();
}

/// <summary>
/// Main thread of session class
/// Repeat the GetNotice command and notify the result to the window.
/// </summary>
private void GetNotice()
{
    while (thread != null)
    {
        string str = SendCommand("GetNotice");
        if (OnGetNotice != null)
        {
            OnGetNotice(str);
        }
    }
}

/// <summary>
/// Obtain images from a camera server.
/// </summary>
public Image GetImage()
{
    string url = server + "GetLiveImage?connection_id=" + connectionId;
    HttpWebRequest req = (HttpWebRequest)HttpWebRequest.Create(url);
    HttpWebResponse res = (HttpWebResponse)req.GetResponse();
    return new Bitmap(res.GetResponseStream());
}

/// <summary>
/// Send a command to a camera server and return the response.
/// </summary>
public string SendCommand(string command)
{
    if (connectionId != null)
    {
        if (command.IndexOf('?') > 0)

```

```

        {
            command = command + "&connection_id=" + connectionId;
        }
        else
        {
            command = command + "?connection_id=" + connectionId;
        }
    }

    try
    {
        command);
        HttpWebRequest req = (HttpWebRequest)HttpWebRequest.Create(server +
        HttpWebResponse res = (HttpWebResponse)req.GetResponse();
        StreamReader sr = new StreamReader(res.GetResponseStream());
        return sr.ReadToEnd();
    }
    catch (Exception){}

    return null;
}

public void Dispose()
{
    thread = null;
}
}
}

```

What this class implements is as follows.

- Obtaining a connection ID using an OpenCameraServer command in the constructor, which starts a session with a camera server.
- Creating a thread and executing a GetNotice command in the thread. Events occurred on the camera server will be notified to listener WvCameraCtrlViewer.
- SendCommand() sends a command to a camera server and then returns the response.
- GetImage() obtains the newest live image from a camera server.

5.3 Combining GUI and Session

Finally, implement the camera control client using this class. First, open the “WvCameraCtrlViewer.cs” file and input as shown below.

WvCameraCtrlViewer.cs
<pre> using System; using System.Collections.Generic; </pre>


```

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.IO;
using System.Windows.Forms;

namespace WvCameraCtrlViewer
{
    /// <summary>
    /// Window class
    ///
    /// Create and lay out a scrollbar for PTZ and a camera control
    /// </summary>
    public partial class WvCameraCtrlViewer : Form
    {
        private WvSession session;
        private Timer imageTimer;
        private delegate void GetNoticeCallback(string str);

        public WvCameraCtrlViewer()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Start a session with a camera server.
        ///
        /// This callback function is executed after the GUI is initialized.
        /// </summary>
        private void WvCameraCtrlViewer_Load(object sender, EventArgs e)
        {
            // 0) Start a session with a camera server.
            session = new WvSession("http://172.20.28.60/-vvhttp-01/");

            // 1) Register the event handler fuction.
            session.OnGetNotice += new WvEventCallback(ProcessServerResponse);

            // 2) Brings the sliderbar and others up to date.
            string str = session.SendCommand("GetCameraInfo");
            ProcessServerResponse(str);

            // 3) Start the timer for obtaining images.
            imageTimer = new System.Windows.Forms.Timer();
            imageTimer.Interval = 500;
            imageTimer.Tick += new EventHandler(ImageTimerTick);
            imageTimer.Enabled = true;
        }
    }
}

```

```

/// <summary>
/// Callback function for the timer
/// </summary>
private void ImageTimerTick(object sender, EventArgs e)
{
    liveScreen.Image = session.GetImage();
}

/// <summary>
/// Interpret an event notification from a camera server and renew the GUI.
/// </summary>
void ProcessServerResponse(string str)
{
    if (string.IsNullOrEmpty(str))
        return;

    if (this.InvokeRequired)
    {
        GetNoticeCallback d = new GetNoticeCallback(ProcessServerResponse);
        this.Invoke(d, new object[] { str });
    }
    else
    {
        int pVal = int.MaxValue, tVal = int.MaxValue, zVal = int.MaxValue;

        StringReader sr = new StringReader(str);
        while (true)
        {
            string line = sr.ReadLine();
            if (line == null) break;

            if (line.StartsWith("pan_current_value") || line.StartsWith("pan="))
            {
                pVal = GetValue(line);
            }
            else if (line.StartsWith("tilt_current_value") || line.StartsWith("tilt="))
            {
                tVal = GetValue(line);
            }
            else if (line.StartsWith("zoom_current_value") || line.StartsWith("zoom="))
            {
                zVal = GetValue(line);
            }
            else if (line.StartsWith("pan_left_limit="))
            {
                pScrollBar.Minimum = GetValue(line);
            }
            else if (line.StartsWith("pan_right_limit="))
            {
                pScrollBar.Maximum = GetValue(line);
            }
        }
    }
}

```

```

else if (line.StartsWith("tilt_up_limit="))
{
    tScrollBar.Maximum = GetValue(line);
}
else if (line.StartsWith("tilt_down_limit="))
{
    tScrollBar.Minimum = GetValue(line);
}
else if (line.StartsWith("zoom_tele_end="))
{
    zScrollBar.Minimum = GetValue(line);
}
else if (line.StartsWith("zoom_wide_end="))
{
    zScrollBar.Maximum = GetValue(line);
}
else if (line.StartsWith("enabled_camera_control"))
{
    controlButton.Text = "control";
    controlButton.BackColor = Color.Green;
}
else if (line.StartsWith("disabled_camera_control") ||
line.StartsWith("failed_to_get_camera_control"))
{
    controlButton.Text = "push";
    controlButton.BackColor = Color.Yellow;
}
else if (line.StartsWith("waiting_camera_control"))
{
    controlButton.Text = "waiting";
    controlButton.BackColor = Color.Magenta;
}
}

// Note: Please note that the tile range is upside down.
if (pVal != int.MaxValue)
    pScrollBar.Value = pVal;
if (tVal != int.MaxValue)
    tScrollBar.Value = tScrollBar.Maximum + tScrollBar.Minimum - tVal;
if (zVal != int.MaxValue)
    zScrollBar.Value = zVal;
}
}

private int GetValue(string line)
{
    int p = line.IndexOf('=');
    if (p > 0)
        return int.Parse(line.Substring(p + 1));
    return 0;
}

```

```

// Callback function called when the control button is pressed.
private void controlButton_Click(object sender, EventArgs e)
{
    session.SendCommand("GetCameraControl");
}

// Callback function called when the scroll bar (pan) is controlled
private void pScrollBar_Scroll(object sender, ScrollEventArgs e)
{
    if (e.Type == ScrollEventType.EndScroll)
        session.SendCommand("OperateCamera?pan="+e.NewValue);
}

// Callback function called when the scroll bar (tilt) is controlled.
private void tScrollBar_Scroll(object sender, ScrollEventArgs e)
{
    if (e.Type == ScrollEventType.EndScroll)
    {
        int t = tScrollBar.Maximum + tScrollBar.Minimum - e.NewValue;
        session.SendCommand("OperateCamera?tilt=" + t);
    }
}

// Callback function called when the scroll bar (zoom) is controlled.
private void zScrollBar_Scroll(object sender, ScrollEventArgs e)
{
    if (e.Type == ScrollEventType.EndScroll)
        session.SendCommand("OperateCamera?zoom=" + e.NewValue);
}

// Callback function called when the window is closed. Close the session.
private void WvCameraCtrlViewer_FormClosing(object sender, FormClosingEventArgs e)
{
    imageTimer.Enabled = false;

    session.OnGetNotice -= new WvEventCallback(ProcessServerResponse);
    session.SendCommand("CloseCameraServer");
    session.Dispose();
}
}

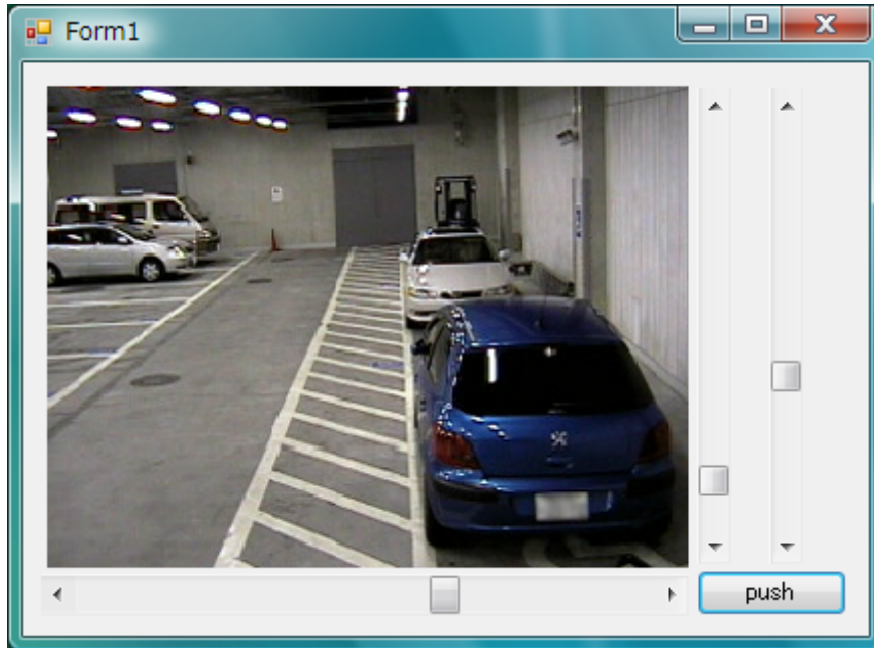
```

What this class implements is as follows.

- Generating an WvSession object after creating GUI.
- Obtaining live images from the WvSession object using a timer and display them.
- Sending commands to a camera server in response to GUI operations.
- Reflecting camera server's events to the GUI, such as a camera position, status

changes of camera control privilege etc.

Choose “Debug” > “Start Debugging” to start the camera control application.



When you create clients using .NET, please heed the following advice.

- GetNotice is a command to receive notifications about status changes from camera servers. The thread issuing GetNotice command might be blocked until an event occurs on the camera server. It means that your program might stop unless you create a GetNotice-specific thread. Please remember that with a Windows Form Application, the application won't exit even if you close the window, unless all threads are exited.

- Windows Form Controls are not thread-safe.

You need to pay attention to which thread renews the GUI, such as renewal of scrollbar's tab position, image drawing, etc. In the above example, we used System.Windows.Forms.Timer to display live video. The Timer object can display images because the object's callback functions ensure access to Windows Form Controls. On the other hand, you cannot access to Windows Form Controls with a self-created thread, System.Threading.Timer, and System.Timers.Timer. In the above application, we created a GetNotice-specific thread to reflect status changes of camera control privilege and changes of PTZ values to the GUI. However, you cannot control Windows Form Controls directly from this thread. In the above example, GetNotice delegates its procedure to the one, which created a control using Control.Invoke. For more details, please refer to the following site.

<http://msdn2.microsoft.com/en-us/library/ms171728.aspx>

6 Conclusion

So far, we have explained the overview of WV-HTTP protocols using several examples. Those are, however, only a part of the whole protocols, as there are many other commands and parameters in WV-HTTP protocols. Once you master the procedures explained in this tutorial, it will be easy for you to master many other commands. We hope this tutorial would help you to build and customize a network camera system.

