Johnson Controls

Digital Vision Network

5000 and 3000 Series

Software Development Kit

(SDK)

Reference Manual

February, 2009
24-10352-18 Revision –

Digital Vision Network

5000 and 3000 Series

Software Development Kit (SDK)

Reference Manual

February, 2009

24-10352-18 Revision –

Johnson Controls

Security Solutions
(805) 522-5555
www.johnsoncontrols.com

Acknowledgment

If this document is translated from the original English version by Johnson Controls, Inc., all reasonable endeavors will be used to ensure the accuracy of translation. Johnson Controls, Inc. shall not be liable for any translation errors contained herein or for incidental or consequential damages in connection with the furnishing or use of this translated material.

Due to continuous development of our products, the information in this document is subject to change without notice. Johnson Controls, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with furnishing or use of this material. Contents of this publication may be preliminary and/or may be changed at any time without any obligation to notify anyone of such revision or change, and shall not be regarded as a warranty.

CE

Declaration of Conformity

This product complies with the requirements of the European Council Electromagnetic Compatibility Directive 2004/108/EEC and the Low Voltage Directive 2006/95/EEC.

This equipment must not be modified for any reason and it must be installed as stated in the Manufacturer's instruction.

If this shipment (or any part thereof) is supplied as second-hand equipment, equipment for sale outside the European Economic Area or as spare parts for either a single unit or system, it is not covered by the Directives.

TABLE OF CONTENTS

Chapter 1: Overview

Chapter 2: Tutorial

Chapter 1

OVERVIEW

The DVN 5000 SDK component (SpectivaWeb) and the DVN 3000 SDK component (ProximaWeb) are two fully customizable ActiveX controls that can interface with Johnson Controls digital video recorder servers (including DVN 5000 and DVN 3000), with the Johnson Controls P2000 Security Management System (SMS)[1], and can be embedded in third party applications developed with common software environments that support ActiveX technology, such as Microsoft® Internet Explorer, Microsoft .NET framework, Microsoft Visual C++, Microsoft Visual Basic, Borland® Delphi, and Borland C++ Builder.

ActiveX controls, formerly known as Object Linking and Embedding (OLE) controls, let you develop sophisticated controls based on the Common Object Model (COM) that can be installed in dialog boxes or any ActiveX control container application, including pages on the Internet's World Wide Web and Visual Basic applications.

An ActiveX control is a COM-based object that can draw itself in its own window, respond to events (such as mouse clicks), and be managed through an interface that includes properties and methods similar to those in Automation objects.

These controls can be developed for many uses, such as database access, data monitoring, or graphing. Besides their portability, ActiveX controls support features previously not available to custom controls, such as compatibility with existing OLE containers and the ability to integrate their menus with the OLE container menus. In addition, an ActiveX control fully supports Automation, which allows the control to expose writable properties and a set of methods that can be called by the control user.

REAL-TIME VIEW AND CONTROL OF CAMERAS

ActiveX controls make it possible to view simultaneously real-time video streams originating from many cameras connected to many different Johnson Controls servers in real time. ActiveX dynamically manages connections and disconnections, so the views of cameras can be changed on the fly with no delay. ActiveX controls present a simple interface, which allows random access (by time, date, and camera) to all recorded video. If Pan/Tilt/Zoom (PTZ) cameras are connected, each accessed camera can be fully controlled for rapid creation of customized programs, allowing for powerful and flexible automation of entire surveillance systems requiring absolutely no human intervention. ActiveX controls make use of the rights assignments available on the connected servers, so user access levels can be maintained at all times.

1. DVN 5000 integration only

REMOTE MANAGEMENT OF ALARMS

When an alarm is triggered on an ActiveX-enabled server, an event notification is sent to the ActiveX application so appropriate responses can be invoked in response (in real time). Built-in Alarm Management and Notification functionality allows for powerful integration between Johnson Controls' Digital Video Management Systems (DVMS) servers and third-party systems via ActiveX.

RECORDING AND TRANSMISSION OF AUDIO CHANNELS

Johnson Controls' DVMSs are equipped with powerful audio recording and transmission features with a built-in audio/video mixer (up to 4 channels). ActiveX controls allow simple creation of audio/video-enabled Windows applications, which requires only a video card and a sound card to receive live and recorded video in real time. ActiveX audio is bi-directional; "talkback" audio can be sent back to the server – all you need is a microphone.

Chapter 2

TUTORIAL

This chapter describes some simplistic approaches to deploy the ProximaWeb ActiveX Control and SpectivaWeb ActiveX Control. As ActiveX Controls can be conveniently used in many different development environments, this chapter includes information necessary to use these components in the most common Integrated Development Environment (IDE).

In addition, this chapter presents a quick tour of more powerful functions.

HANDS-ON – BORLAND DELPHI AND C++ BUILDER

Importing the ActiveX Control

You can import the **Spectiva ActiveX Control** or **Proxima ActiveX Control** in Delphi as described in the Delphi documentation, which is partially duplicated here:

1. From the menu, select **Component>Import ActiveX Control**.
2. Select the **Spectiva Web Control Module** type library from the list or **Johnson Controls, Inc Proxima ActiveX Control module**.

   The dialog lists all the registered libraries that define ActiveX controls (this is a subset of the libraries listed in the Import Type Library dialog). If the type library is not in the list, click the **Add** button, find and select the S**pectivaWeb.ocx** or **ProximaWeb.ocx** control file, and click **OK**. This registers the type library, making it available. Then repeat step 2.

   You can then click the **Install** button.
3. If you do not want to install the ActiveX control on the Component palette, select **Create Unit**. This generates the *TypeLibName_TLB* unit and adds the declaration of its component wrapper. This exits the Import ActiveX dialog.
4. To install the ActiveX control on the Component palette, select the Palette page on which this component will reside and click **Install**. This generates the *TypeLibName_TLB* unit, like the **Create Unit** button, and displays the Install component dialog, letting you specify the package where the components should reside (either an existing package or a new one).

   After exiting the Import ActiveX dialog, the new *TypeLibName_TLB* unit appears in the directory specified by the **Unit dir name** control. This file contains declarations for the elements defined in the type library, as well as the generated component wrapper for the ActiveX control.

If you installed the generated component wrapper, an ActiveX control now resides on the Component palette. You can use the Object Inspector to set properties or write event handlers for this control.

HANDS-ON – JAVASCRIPT

Hosting the ActiveX

The first pass to handle a connection with a remote server is instantiate an ActiveX object in the HTML page. You can enter the following code in the body section of your page:

```
<OBJECT ID="VisionWeb1" WIDTH=345 HEIGHT=282
CLASSID="CLSID:10458b03-35ac-4d5c-b9aa-9645f27b3e4d"
codebase="ProximaVisionWEB.cab#version=3,0,0,27">
```

HANDS-ON – MICROSOFT VISUAL BASIC 6

Importing the ActiveX Control

You can import the **Spectiva ActiveX Control** or **Proxima ActiveX Control** in Visual Basic as described in the Microsoft Developer Network (MSDN) documentation, which is partially reproduced here:

1. On the **Project** menu, click **Components** to display the Components dialog box.

   Items listed in this dialog box include all registered, insertable objects, designers, and ActiveX controls.
2. To add an ActiveX control to the Toolbox, select the check box to the left of the control name.
3. Click **OK** to close the Components dialog box. The selected ActiveX controls appear in the Toolbox.

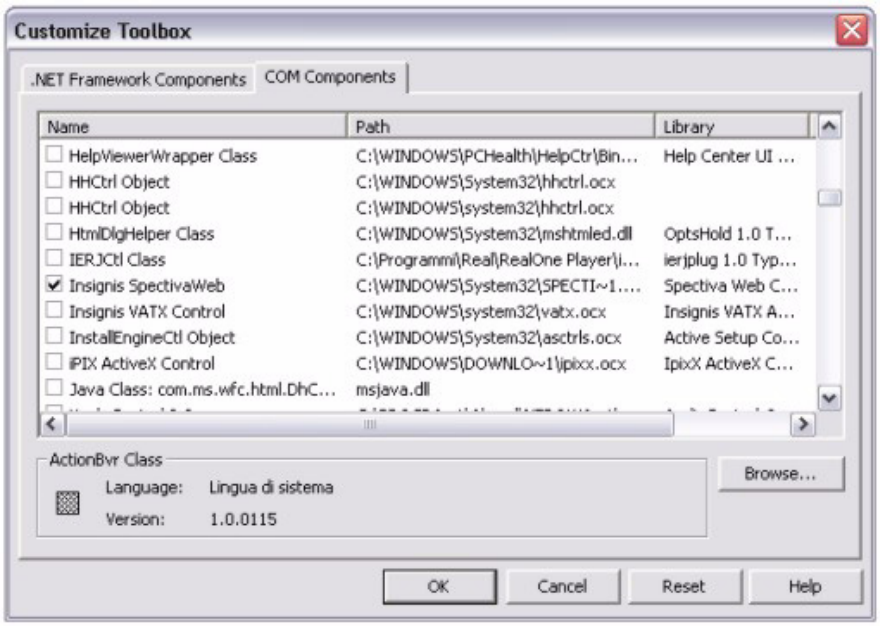
To add ActiveX controls to the Components dialog box, click the **Browse** button and locate the **ProximaWeb.ocx** and/or **SpectivaWeb.ocx** files. These files are commonly installed in the *Windows\System* or *Windows\System32* directory. When you add an ActiveX control to the list of available controls, Visual Basic automatically selects its check box in the Components dialog box.

HANDS-ON – MICROSOFT .NET ENVIRONMENT AND VISUAL C++

Importing the ActiveX Control

You can import the **Spectiva ActiveX Control** or **Proxima ActiveX Control** in .NET as described in the MSDN documentation, which is partially reproduced here:

1. Create your new .NET or Visual C++ MFC application.
2. Right-click on the ToolBox and select **Add/Remove items**.
3. Select the **COM Components** tab.
4. Select **Insignis Spectiva Web** or **Johnson Controls, Inc VisionWeb** ActiveX controls.



5. If these controls are not present in the COM Components list, click the **Browse** button and select **SpectivaWeb.ocx** or **ProximaWeb.ocx**.
6. A new graphic control will be inserted in the ToolBox.

METHODS

The following methods are provided by the dispatch interface of the Proxima and Spectiva ActiveX controls. The following methods are listed in alphabetical order; however, provided SDK functions are grouped into the following categories:

- Connection
- Windows Management
- Playback
- AUX and Alarms Management
- Audio
- Dome Piloting
- Spot Monitors
- Export
- Text Insertion
- Manual Recording

CONNECTION METHODS

This group contains the methods necessary to establish, manage and close a connection to a video server. These methods are:

- Connect (see page 2-5)
- CryptConnect (see page 2-6)
- Disconnect (see page 2-7)
- DeltaTransmission (see page 2-8)
- RequestStreamInfo (see page 2-9)
- SendImages (see page 2-10)
- SendCertificate (see page 2-11)
- SetConnectionPriority (see page 2-12)

Connect Method

Purpose

Connect to a DVR server by specifying the hostname or IP address, and the delta type, user name, and password.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long Connect(BSTR *_user*, BSTR *_passwd*, BSTR *_host*, long *_delta*, short *_keyFrameRefresh*, short *_bitRate*);

Input Parameters

Table 2-1: Input Parameters

Type	Name	Description
BSTR	_user	Name of user for server login
BSTR	_passwd	User password
BSTR	_host	Hostname or IP address of DVMS server to connect to
long	_delta	Boolean: 0 to deactivate delta transmission. 1 to activate.
short	_keyFrameRefresh	Used only by DVN 3000 Servers using Delta; sets the maximum distance between two keyframes.
short	_bitRate	Used only by DVN 3000 Servers using Delta; attempts to maintain an average bit rate.

Return

unique ID for the server connection.

-1 if a generic error occurred during connection, or maximum supported connection limit already reached.

Remarks

When using this method to connect to a DVN 5000 server, encryption is automatically used. To use password encryption connecting to a DVN 3000 server, use the *CryptConnect* method.

The action of the *Connect()* method depends on the values supplied for its parameters. Specifically, if the **IP address** the caller passes is null or blank, the SDK interprets this as a request to save the **username** and **password** in shared memory for use in later calls and to delay any attempt to connect to the DVN. If the username and password the caller passes are null or blank, the SDK interprets this as a request to retrieve the username and password from shared memory and then attempt to connect to the DVN.

Also, if a hostname rather than an IP address is supplied, then the computer on which the SDK call is being made must have access to a Domain Name System (DNS) that can resolve the hostname, and the DVN's name (as defined on RemoteControl's System Configuration screen by clicking the **Network Cards** button) must be set to that hostname (see the *DVN 5000 Software Installation and Configuration Manual* for detailed information on configuring the DVN 5000 server's name).

See Also

- "CryptConnect Method" on page 2-6
- "Disconnect Method" on page 2-7

CryptConnect Method

Purpose

Connect to a DVR server specifying the IP address, delta type, user name, and password using a strong encryption model.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long Connect(BSTR *_user*, BSTR *_passwd*, BSTR *_host*, long *_delta*, short *_keyFrameRefresh*, short *_bitRate*);

Input Parameters

Table 2-2: Input Parameters

Type	Name	Description
BSTR	_user	Name of user for server login
BSTR	_passwd	User password
BSTR	_host	IP address of DVMS server to connect to
long	_delta	Boolean: 0 to deactivate delta transmission. 1 to activate.
short	_keyFrameRefresh	Used only by DVN 3000 Servers using Delta; it sets the maximum distance between 2 keyframes
short	_bitRate	Used only by DVN 3000 Servers using Delta; attempts to maintain an average bit rate.

Return

unique ID for the server connection.

-1 if a generic error occurred during connection.

-2 if the specified server type is not valid.

See Also

- “Connect Method” on page 2-5
- “Disconnect Method” on page 2-7

Remarks

This function is only supported by DVN 3000 Version 3.0 or later. To connect to a DVN 3000 video server or to a DVN 3000 beta or Release Candidate (RC), only the Connect method is supported.

Disconnect Method

Purpose

Closes the connection to the specified DVR server.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long Disconnect (long _position)

Input Parameters

Table 2-3: Input Parameters

Type	Name	Description
long	_ position	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* methods.

Return

-1 if the given connection ID is not valid or it is a local video footage playback connection.

1 on success.

See Also

- “Connect Method” on page 2-5
- “CryptConnect Method” on page 2-6

DeltaTransmission Method

Purpose

Instructs the DVR Server to activate or deactivate delta video transmission for the selected connection.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long DeltaTransmission(long _activate, long _position)

Input Parameters

Table 2-4: Input Parameters

Type	Name	Description
long	_activate	Boolean: Set to 0 for normal wavelet transmission. Set to 1 for delta-wavelet.
long	_position	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* methods.

Returns

1 on success.

-1 if the given connection ID is not valid

Remarks

The delta transmission algorithm depends on the version of the software installed on the video server:

- DVN 3000 uses DeltaWLT, RC1 to 3.00.02 uses DeltaHF, and Version 3.00.03 and above uses DeltaV4 (Enpacta)
- DVN 5000 uses DeltaHF and DeltaV4 (Enpacta)

RequestStreamInfo Method

Purpose

Instructs the DVR server to periodically send basic stream information about the video shown in the specified window to the client.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long RequestStreamInfo(long window, long* date, long interval);

Input Parameters

Table 2-5: Input Parameters

Type	Name	Description
long	window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long*	date	Pointer to an **__int64** containing the date of the frame currently viewed in the specified window.
long	interval	Period, expressed in milliseconds, of the *OnStreamInfo* event used to return the requested information; use 0 to stop.

Return

1 on success

0 if **window** parameter points to an invalid or non-existing window.

Remarks

If the **interval** parameter is greater then 0, the video server tries to post the required information to the client every **interval** milliseconds, but this period could increase in case of heavy workload.

When this information is received, an *OnStreamInfo* event is triggered.

See Also

- “OnStreamInfo Event” on page 2-101
- “SetExternalWin Method” on page 2-19

SendImages Method

Purpose

Orders the DVR Server to Start or Stop sending video streams to the client.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SendImages(long _connection, long _status)

Input Parameters

Table 2-6: Input Parameters

Type	Name	Description
long	_ connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* methods.
long	_status	Boolean: Set to 1 to **start** the image transmission. Set to 0 to **stop** the image transmission.

Return

1 on success.

-1 if the given connection ID is not valid or if it is an ID of a refused connection.

SendCertificate Method

Purpose

SpectivaWeb sends code calls for an answer method to the *OnNeedCertificate Event*, which passes to the client an encrypted buffer to be decrypted and sent back.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long SendCertificate(long * buffer, long size, long connection)

Input Parameters

Table 2-7: Input Parameters

Type	Name	Description
long*	buffer	A buffer containing decrypted data to be sent back to the server as user validation.
long	size	Size in bytes of the data structure pointed at ***buffer**.
long	connection	ID of server connection.

Return

1 on success.

-1 if the given connection ID is not valid or if it is an ID of a refused connection.

SetConnectionPriority Method

Purpose

Sets the priority of the thread that manages the connection with the server. As this thread owns, among the other things, the messages dispatcher, increasing this thread priority could increase ActiveX reactivity on incoming events.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long SetConnectionPriority(long connection, long priority)

Input Parameters

Table 2-8: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* methods.

Table 2-8: Input Parameters

Type	Name	Description
long	priority	Valid range: [0..2] 0 = above normal priority 1 = highest priority 2 = time critical priority

Return

1 on success.

0 if the given connection ID is not valid or if it is an ID of a refused connection.

Remarks

Suggested values are:

0 for slow machines (2 GHz or slower)

1 for mid-range machines (2 to 3 GHz)

2 for high range machines (3 GHz and faster) and only in applications where real-time events management is strictly required. Avoid setting the priority as time critical for more than one connection at a time.

WINDOWS MANAGEMENT METHODS

This group contains all methods necessary to manage window size and resolution, to map Windows Handle to internal references, and to prepare a grid, useful for quad and custom display. The Windows Management methods are:

- CreateGrid (see page 2-14)
- CreateWin (see page 2-15)
- SetCamera (see page 2-16)
- ChangeResolutionWin (see page 2-17)
- ResizeWindow (see page 2-18)
- SetExternalWin (see page 2-19)
- DestroyExternalWin (see page 2-20)
- ChangeHwndWin (see page 2-21)

CreateGrid Method

Purpose

Splits the ActiveX client area into a grid of **_ncol** columns and **_nrow** rows, allowing many different video streams to display in the same window.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long CreateGrid(short _ncol, short _nrow);

Input Parameters

Table 2-9: Input Parameters

Type	Name	Description
short	_ncol	Number of columns. Valid range [0..12].
short	_nrow	Number of rows. Valid range [0..12].

Return

1 on success

Remarks

If **_ncol** or **_nrow** is out of the valid range, it is reduced to the nearest valid value. Once the grid is created, video display windows must be created within the grid to display video streams. To create these windows within this grid, use the *CreateWin* method.

See Also

“CreateWin Method” on page 2-15

CreateWin Method

Purpose

Create a display window within a grid of an ActiveX client area. Use the *CreateGrid* method to create this grid.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long CreateWin(short _x1, short _y1, short _x2, short _y2)

Input Parameters

Table 2-10: Input Parameters

Type	Name	Description
short	_x1	Abscissas coordinates of the upper left corner of the client window
short	_y1	Ordinates coordinates of the upper left corner of the client window
short	_x2	Abscissas coordinates of the lower right corner of the client window
short	_y2	Ordinates coordinates of the lower right corner of the client window

Return

unique ID of the window created

-1 if the grid was not initialized or if the MAX_WINDOWS limit has been reached.

Remarks

Before creating a window using *CreateWin*, initialize the grid using the *CreateGrid* method.

Abscissas and Ordinates coordinates **are not** pixel references; they are numbers of columns and rows in the grid.

See Also

“CreateGrid Method” on page 2-14

Example

Creation of a grid and grid setup for quad display.

```
VisionWeb1->CreateGrid(12,12);
win4_1 = VisionWeb1->CreateWin(0,0,6,6);
win4_2 = VisionWeb1->CreateWin(6,0,12,6);
win4_3 = VisionWeb1->CreateWin(0,6,6,12);
win4_4 = VisionWeb1->CreateWin(6,6,12,12);
```

SetCamera Method

Purpose

Displays video streams from a camera of the specified connection in the window passed as parameter, or removes the camera from the window.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

SetCamera(short _numTele, short _connection, long _window)

Input Parameters

Table 2-11: Input Parameters

Type	Name	Description
short	_numTele	ID of camera
short	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

Remarks

Use -1 as **_numTele** parameter to remove the camera display from the specified window.

ChangeResolutionWin Method

Purpose

Instructs the DVR server to switch video resolution of the video displayed in the window between QCIF, CIF, 3/4 FULL and FULL resolution.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long ChangeResolutionWin(long _resolution, long _win);

Input Parameters

Table 2-12: Input Parameters

Type	Name	Description
long	_resolution	Constant indicating the chosen resolution. Valid range [0..3] See “Remarks” on page 2-17 for constant to pixels resolution map.
long	_win	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success

-1 if a generic error occurred.

Remarks

Table 2-13 shows **_resolution** values and image resolutions.

Table 2-13: _resolution Values and Image Resolutions

_resolution	Width	NTSC Height	PAL Height
NORMAL_RESOLUTION = **0**	360	243	288
CIF_RESOLUTION = **1**	176	122	144
QCIF_RESOLUTION = **2**	90	62	72
MAX_RESOLUTION = **3**	720	243	288

Delta connections **do not** support MAX_RESOLUTION.

If *ChangeResolutionWin* is called with MAX_RESOLUTION for a window belonging to a connection using a delta transmission, it will use NORMAL_RESOLUTION instead.

If a DVR server uses DeltaWavelet, no more than **one** window for each connection can use NORMAL_RESOLUTION; all the other windows will use CIF_RESOLUTION. This limit does not exist if the DVR server uses Enpacta as the delta transmission protocol.

When the DVR server is using hardware compression, this function has no effect on cameras recorded in Enpacta or MPEG4.

ResizeWindow Method

Purpose

Changes the size of the window rectangle used to display video streams.

Compatibility

DVN 3000: Yes

DVN 5000: No (automatically detected)

Syntax

ResizeWindow(long _window, long _x, long _y, long _width, long _height)

Input Parameters

Table 2-14: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Table 2-14: Input Parameters

Type	Name	Description
long	_x	No more used; suggested value: 0
long	_y	No more used; suggested value: 0
long	_width	Pixel width of the display rectangle.
long	_height	Pixel height of the display rectangle.

Return

0 on success

Remarks

These parameters represent the real width and height used for the video rendering. They are not necessarily connected to the video resolution.

A low resolution video stream could be displayed at 800x600 if the following methods are called:

```
#define QCIF_RESOLUTION 2
VisionWeb1->ChangeResolutionWin (QCIF_RESOLUTION, windowID);
VisionWeb1->ResizeWindow (windowID, 0, 0, 800, 600);
```

Nevertheless, keep video resolution and window size as congruent as possible.

SetExternalWin Method

Purpose

Sets a generic window (not belonging to the ActiveX) as a display window to show a video stream of a specified connection.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SetExternalWin(long _handle, long _position);

Input Parameters

Table 2-15: Input Parameters

Type	Name	Description
long	_handle	Real window HWND
long	_position	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

unique ID for the window.

-1 if a generic error occurred.

DestroyExternalWin Method

Purpose

Removes the specified window from the list of windows used by the ActiveX.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long DestroyExternalWin (long _handle);

Input Parameters

Table 2-16: Input Parameters

Type	Name	Description
long	_handle	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if the specified window **does not** exist in the given connection.

-2 if the specified window does exist, but **is not** an external window.

Remarks

The **_handle** parameter **is not** the real windows handle, but the internal window ID.

ChangeHwndWin Method

Purpose

Replaces the real window HWND associated with an internal window ID with an HWND belonging to another window.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long ChangeHwndWin (long _handle, _long window);

Input Parameters

Table 2-17: Input Parameters

Type	Name	Description
long	_handle	Target window handle
long	_window	Internal ID of the window whose HWND will be changed.

Return

1 on success.

-1 if the specified window **does not** exist in the given connection.

Remarks

This allows a camera display to move from one window to another without the need to create a new window or set another existing window as an ActiveX external window.

PLAYBACK METHODS

This group contains all methods necessary to playback recorded video footage. These methods are:

You can also view the following methods in the "Connection Methods" section:

Play Method

Purpose

Instructs the DVR server to start sending recorded video for the camera actually viewed in the specified window. It also starts multiple playback if the window belongs to a DVN 5000 server connection for which multiple playback has been configured.

Compatibility

DVN 3000: Yes (without Multiple Playback)

DVN 5000: Yes

Syntax

Play (long _window)

Input Parameters

Table 2-18: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

Remarks

See "AddToMPList Method" on page 2-33 for important remarks about multiple playback.

Stop Method

Purpose

Instructs the DVR server to stop sending recorded video and switch back to live video.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

Stop (long _window)

Input Parameters

Table 2-19: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

PlayBegin Method

Purpose

Instructs the DVR server to move the playback to the beginning of the recording of the active camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

PlayBegin (long _window)

Stop (long _window)

Input Parameters

Table 2-20: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayBegin()* method can only be called **after** the *Play()* method has been called and no error was returned.

PlayEnd Method

Purpose

Instructs the DVR server to move the playback to the end of the recording of the active camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

PlayEnd (long _window)

Input Parameters

Table 2-21: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayEnd()* method can only be called **after** the *Play()* method has been called and no error was returned.

ChangeSectorWin Method

Purpose

Changes the sector displayed on the specified window when in playback mode.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long ChangeSectorWin(long _sector, long _window);

Input Parameters

Table 2-22: Input Parameters

Type	Name	Description
long	_sector	Number of the sector to be displayed.

Table 2-22: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

Remarks

Use following constants value for **_sector**:

Table 2-23: Sector Values

Sector	Value
PRIME	0
TIME LAPSE	1
ALARMS	2

PlayFind Method

Purpose

Instructs the DVR server to move the playback at the given date and hour expressed using the **time_t date/time** data type.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long PlayFind(long _window, long _datetime)

Input Parameters

Table 2-24: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	_datetime	Date and Hour to playback. **_datetime** must be a valid **time_t** value.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayFind()* method can only be called **after** the *Play()* method has been called and no error was returned.

If no recorded video is located at the specified date and hour, the DVN 5000 plays the video with the nearest, subsequent date and time.

PlayFind64 Method

Purpose

Instructs the DVR server to move the playback to the specified date and hour expressed using the **_int64** date/time data type.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long PlayFind64(long _window, long _lowPart, long _highPart)

Input Parameters

Table 2-25: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	_ lowPart	The 32 less significant bits of the **_int64** date/time.
long	_highPart	The 32 most significant bits of the **_int64** date/time.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayFind()* method can only be called **after** the *Play()* method has been called and no error was returned.

If no recorded video is located at the specified date and hour, the DVN 5000 plays the video with the nearest, subsequent date and time.

This method supports random playback re-positioning, even when using Multiple Playback.

PlayStill Method

Purpose

Instructs the DVR server to pause the playback of recorded video.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

PlayStill (long _window)

Input Parameters

Table 2-26: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayStill()* method can only be called **after** the *Play()* method has been called and no error was returned.

When the video playback is paused, the audio playback is also stopped.

PlayFfw Method

Purpose

Start playing in fast forward mode.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long PlayFfw(long _window, long _speed)

Input Parameters

Table 2-27: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	_speed	Valid range [0 ... 512] power of 2. Any other value is rounded down to the nearest power of 2.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayStill()* method can only be called **after** the *Play()* method has been called and no error was returned.

If _speed is 0, the playback is paused.

If _speed is 1, the playback is set at normal speed.

PlayRew Method

Purpose

Starts playing backwards.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long PlayRew(long _window, long _speed)

Input Parameters

Table 2-28: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	_speed	Valid range [0 ... 512] power of 2. Any other value is rounded down to the nearest power of 2.

Return

1 on success.

-1 if a generic error occurred.

Remarks

This method requires the window to be in playback mode, so the *PlayStill()* method can only be called **after** the *Play()* method has been called and no error was returned.

If _speed is 0, the playback is paused.

If _speed is 1, the playback is set at normal speed.

PlayLocal Method

Purpose

Starts to playback an external file containing exported video footage in PSV format or a still image in SSF format.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long PlayLocal(BSTR _nameString)

Input Parameters

Table 2-29: Input Parameters

Type	Name	Description
BSTR	nameString	Full path of image or video to play back.

Return

1 on success.

-1 if the file was not found.

-2 Error while reading header or header CRC error. Probably an unsupported file.

AddToMPList Method

Purpose

Adds a window to the multiple playback windows list.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long AddToMPList(long _window)

Input Parameters

Table 2-30: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Remarks

Multiple synchronized playback works using a list of windows that can be viewed at the same time, keeping the playback time of each window congruent with the hour and date of a particular window (the Master Window).

Before being started, multiple playback requires the completion of the following steps:

1. The multiple playback windows list must be prepared before calling the *Play()* method. This list can be populated with one bare element, but it cannot be empty. Other windows can be attached to the list even after the playback has started. This list can be populated by calling the *AddToMPList()* method. When creating this list for the first time, we suggest clearing it by calling the *ClearMPList()* method.
2. Before calling the *Play()* method, a master window must be set by calling the *SetMPMaster()* method. The window set as master must be present in the multiple playback windows list, otherwise an error will be returned and multiple playback will not start.

Since both single camera playback and multiple camera playback are started by calling the *Play()* method, to switch from multiple playback to single playback, clear the window list using *ClearMPList()*.

Multiple synchronized playback is a cross-connection function, so many different windows belonging to different connections (but always connections to DVN 5000 servers) can be added to the multiple playback windows list.

NOTE

Each instance of the ActiveX control supports only one multiple playback list.

Even when playback has already started, the master window can be changed simply by calling the *SetMPMaster()* method. However, if an invalid window is set as the master (because it does not exist or is not present in the multiple playback windows list), an *OnWinStatusChange* event will be triggered and the parameter **_reason** will probably be MASTER_REMOVED. This event will also be triggered if the master window is removed from the windows list.

See Also

- "Play Method" on page 2-23
- "RemoveFromMPList Method" on page 2-35
- "ClearMPList Method" on page 2-36
- "SetMPMaster Method" on page 2-37
- "GetMPMaster Method" on page 2-38
- "GetMPList Method" on page 2-39
- "OnWinStatusChange Event" on page 2-107

RemoveFromMPList Method

Purpose

Removes a window from the multiple playback windows list.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long RemoveFromMPList(long _window)

Input Parameters

Table 2-31: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if the selected window is invalid.

Remarks

If the removed window is the master window of the multiple playback, an *OnWinStatusChange* event will be triggered and the parameter **_reason** will probably be MASTER_REMOVED.

See “AddToMPList Method” on page 2-33 for important remarks about multiple playback.

- See Also
- “Play Method” on page 2-23
- “AddToMPList Method” on page 2-33
- “ClearMPList Method” on page 2-36
- “SetMPMaster Method” on page 2-37

- "GetMPMaster Method" on page 2-38
- "GetMPList Method" on page 2-39
- "OnWinStatusChange Event" on page 2-107

ClearMPList Method

Purpose

Removes any window from the multiple play windows list.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long ClearMPList()

Input Parameters

None.

Return

1 on success.

Remarks

Call this method to switch from multiple playback to single camera playback. We also suggest clearing the list before inserting the first camera in the list.

See "AddToMPList Method" on page 2-33 for important remarks about multiple playback.

See Also

- "Play Method" on page 2-23
- "AddToMPList Method" on page 2-33
- "RemoveFromMPList Method" on page 2-35
- "SetMPMaster Method" on page 2-37
- "GetMPMaster Method" on page 2-38

■ “GetMPList Method” on page 2-39

SetMPMaster Method

Purpose

Sets a camera in the multiple playback windows list as the Master camera.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long SetMPMaster(long _window)

Input Parameters

Table 2-32: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

Return

1 on success.

-1 if the selected window is invalid.

Remarks

A master camera must be set before starting multiple playback. If the multiple playback windows list is not empty, but no master has been set, the system will not be able to start playback (single or multiple).

See “AddToMPList Method” on page 2-33 for important remarks about multiple playback.

See Also

- "Play Method" on page 2-23
- "AddToMPList Method" on page 2-33
- "RemoveFromMPList Method" on page 2-35
- "ClearMPList Method" on page 2-36
- "GetMPMaster Method" on page 2-38
- "GetMPList Method" on page 2-39

GetMPMaster Method

Purpose

Retrieves the master camera in the multiple playback windows list.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long GetMPMaster()

Input Parameters

None.

Return

The index of the master window, on success.

-1 if the master camera was not found.

Remarks

See "AddToMPList Method" on page 2-33 for important remarks about multiple playback.

See Also

- "Play Method" on page 2-23
- "AddToMPList Method" on page 2-33
- "RemoveFromMPList Method" on page 2-35
- "ClearMPList Method" on page 2-36
- "GetMPMaster Method" on page 2-38
- "GetMPList Method" on page 2-39

GetMPList Method

Purpose

Fills a pre-allocated array of window indexes present in the multiple playback windows list.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long GetMPList (long FAR* _winHandles, long FAR* _count)

Input Parameters

Table 2-33: Input Parameters

Type	Name	Description
long FAR*	_winHandles	Pointer to an empty array of (_count * sizeof(long)) byte.
long FAR*	_count	Number of DWORD

Return

1 on success.

-1 if the array was not correctly allocated.

Remarks

The array **is not** filled with real Windows handlers, but they are the internal window indexes.

See “AddToMPList Method” on page 2-33 for important remarks about multiple playback.

See Also

- “Play Method” on page 2-23
- “AddToMPList Method” on page 2-33
- “RemoveFromMPList Method” on page 2-35
- “ClearMPList Method” on page 2-36
- “GetMPMaster Method” on page 2-38
- “SetMPMaster Method” on page 2-37

GetFindData Method

Purpose

Instructs the DVR server to send **Find** data.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long GetFindData(long connection, long cameras, long sector, long data);

Input Parameters

Table 2-34: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Table 2-34: Input Parameters

Type	Name	Description
long	cameras	A bit-field indicating the camera from which data will be located. Use only the 16 less significant bits: Bit 1 = camera 1, Bit 2 = camera 2, and so on.
long	sector	Number of the sector to be played back. See “ChangeSectorWin Method” on page 2-26 for Constants values.
long	data	Pointer to a *FindDataStruct* structure (see “Remarks” on page 2-41 for details)

Return

1 on success.

Remarks

To start a new find session, invoke a *GetFindData* method with a *findDataStruc* structure and a **sendData** field of 2.

```
struct FindDataStruct
{
    __int64 startTime;
    __int64 endTime;
    __int64 seconds; // time filter: Intervals far each other less than this are
                     // considered as one bare interval.
    unsigned long int sendData; // sendData = 2 for initial data request.
};
```

When required data is ready to be sent to the client, an *OnFindData* event is triggered. Once a request has been sent, the first *OnFindData* event responds with a **FindDataStruct** containing precise data for each field. This same structure can be used to forward a new *GetFindData* request, whose reply event will be another *OnFindData* event, but this time it will contain a buffer with *n* pairs of **__int64**, where *n* is the number of recorded intervals.

See Also

“OnFindData Event” on page 2-100

ConvertImageToBitmap Method

Purpose

Converts the last received Wavelet image to bitmap format and provides information about the converted image.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long ConvertImageToBitmap(long* _destBuffer, long* _date, long* _numSector, long* _numCamera, long* _local, long _connection);

Input Parameters

Table 2-35: Input Parameters

Type	Name	Description
long*	_destBuffer	Return parameter. A buffer containing the last received Wavelet image converted to bitmap format.
long*	_date	Return parameter. A **time_t** representing the image date and hour.
long*	_numSector	Return parameter. Number of the sector to which the image belongs.
long*	_numCamera	Return parameter. Camera number of the converted image.
long*	_local	Return parameter. 0 for remote connection. 1 for local playback.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

ConvertImageToBitmapWin Method

Purpose

Converts the Wavelet image shown in the selected window to bitmap format and provides information about the converted image.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long ConvertImageToBitmapWin(long* _destBuffer, long* _date, long* _numSector, long _window, long* _numCamera, long* _local, long _connection)

Input Parameters

Table 2-36: Input Parameters

Type	Name	Description
long*	_destBuffer	Return parameter. A buffer containing the last received Wavelet image converted to bitmap format.
long*	_date	Return parameter. A **time_t** representing the image date and hour.
long*	_numSector	Return parameter. Number of the sector to which the image belongs.
long	_window	Number of the window showing the image you wish to convert to bitmap format.
long*	_numCamera	Return parameter. Camera number of the converted image.
long*	_local	Return parameter. 0 for remote connection. 1 for local playback.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

EnableVideoQualityEnhanceWin Method

Purpose

Improves image quality by removing MPEG4 and Enpacta compression artifacts.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long EnableVideoQualityEnhanceWin(long connection, long window, long status)

Input Parameters

Table 2-37: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	status	Quality of the post processing.

Return

1 on success.

Remarks

If the **window** parameter is set to -1, post processing is applied to all the windows belonging to the selected connection.

If the **window** parameter **is not** set to -1, the connection parameter is useless, so it is ignored.

If the **connection** and **window** parameters are set to -1, quality enhancement is applied to all the windows.

Valid values for the status parameter are:

- noVideoQualityEnhanceStatus = 0
- VideoQualityEnhanceStatusLow = 1
- VideoQualityEnhanceStatusMedium = 2
- VideoQualityEnhanceStatusHigh = 3

A higher processing status parameter equals higher quality. However, more CPU resources are used.

The default status is 0.

When using Wavelet or Enpacta as the compression codec, there is no difference between *VideoQualityEnhanceStatusLow*, *VideoQualityEnhanceStatusMedium*, and *VideoQualityEnhanceStatusHigh*.

EnableMpegD1QualityEnhanceWin Method

Purpose

Improves image quality by removing unwanted motion artifacts.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long EnableMpegD1QualityEnhanceWin(long connection, long window, long status)

Input Parameters

Table 2-38: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	status	Status of the quality improvements. Set status to 0 to enable. Set status to 1 to disable.

Return

1 on success.

Remarks

If the **window** parameter is set to -1, post-processing is applied to all the windows belonging to the selected connection.

If the **window** parameter **is not** set to -1, the connection parameter is useless, so it is ignored.

If the **connection** and **window** parameters are set to -1, quality enhancement is applied to all the windows.

AUX AND ALARM METHODS

This group contains all methods necessary to set and retrieve information to and from AUXes and Alarms. These are:

- SendAlarm (see page 2-46)
- SendAux (see page 2-47)
- SendCameraAux (see page 2-48)
- ReceiveStatusAuxAlarm (see page 2-49)

SendAlarm Method

Purpose

Notifies the DVR server to turn on or off the specified alarm.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SendAlarm(long _connection, long _numAlarm, long _status)

Input Parameters

Table 2-39: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	_numAlarm	Number of the alarm to turn on or off. Valid range [0 .. 16]
long	_status	Boolean: Set to 1 to turn **on** the alarm. Set to 0 to turn **off** the alarm.

Return

1 on success.

-1 if a generic error occurred.

SendAux Method

Purpose

Notifies the DVR server to turn on or off the specified AUX.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SendAux(long _connection, long _numAux, long _status)

Input Parameters

Table 2-40: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Table 2-40: Input Parameters

Type	Name	Description
long	_numAux	Number of the AUX to turn on or off. Valid range[0 .. 16]
long	_status	Boolean: Set to 1 to turn **on** the AUX. Set to 0 to turn **off** the AUX.

Return

1 on success.

-1 if a generic error occurred.

SendCameraAux Method

Purpose

Notifies the DVR server to turn on or off the specified AUX of the specified camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SendCameraAux(long connection, long, camera, long numAux, long status)

Input Parameters

Table 2-41: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	camera	Number of the camera whose AUX has to be set.
long	numAux	Number of the AUX to turn on or off. Valid range[0 .. 16]

Table 2-41: Input Parameters

Type	Name	Description
long	status	Boolean: Set to 1 to turn **on** the AUX. Set to 0 to turn **off** the AUX.

Return

1 on success.

-1 if a generic error occurred.

ReceiveStatuAuxAlarm Method

Purpose

Instructs the DVR server to send the activation status of AUX and alarms.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long ReceiveStatusAuxAlarm (long _connection);

Input Parameters

Table 2-42: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

-1 if a generic error occurred.

Remarks

As the DVR server finishes collecting the required information, an *OnAuxAlarmStatus* event is triggered and a 32-bit bit-field is returned, in which the 16 less significant bits contain the alarms status, while the most significant bits contain the AUX status.

AUDIO MANAGEMENT METHODS

This group contains all methods necessary to set audio channels settings. These are:

- SetAudioVolume (see page 2-50)
- SetBidirectionalAudioVolume (see page 2-51)

SetAudioVolume Method

Purpose

Sets the volume of the desired audio channel in the specified connection.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SetAudioVolume(long _connection, long _volume, long _channel)

Input Parameters

Table 2-43: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	_volume	Volume value to set for the audio channel. Valid range [0 ... 100]; 0 = turn off.
long	_channel	Number of the audio channel whose volume will be set.

Return

1 on success.

-1 if a generic error occurred.

Remarks

When connecting to a DVR server whose audio channel support is limited to 1, the parameter **_channel** must always be set to 0.

SetBidirectionalAudioVolume Method

Purpose

Sets the volume of the TALK channel from client to server.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SetBidirectionalAudioVolume(long _connection, long _volume);

Input Parameters

Table 2-44: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	_volume	Volume to be set for the TALK channel. 0 = turn off. Valid Range [0 .. 100]
long	_channel	Number of the audio channel whose volume will be set.

Return

0 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 Error creating audio recorder: hardware failure?

Remarks

Depending on the audio board, sometimes this method works as a binary switch to simply turn on or off the TALK channel. In this case, the talk channel is turned off for volume value 0, while it could be active with the maximum volume for any other value.

DOME PILOTING METHODS

This group contains all methods necessary to pilot dome cameras. These methods are:

- MoveDome (see page 2-52)
- ZoomDome (see page 2-53)
- IrisDome (see page 2-54)
- FocusDome (see page 2-55)
- SelectPresetDome (see page 2-56)
- SelectTourDome (see page 2-57)

MoveDome Method

Purpose

Moves camera dome on X and/or Y axis.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long MoveDome(long _xSpeed, long _ySpeed, long _camera, long _connection);

Input Parameters

Table 2-45: Input Parameters

Type	Name	Description
long	_xSpeed	Speed of dome movement on X (horizontal) axis. Valid range: [-100 .. +100]

Table 2-45: Input Parameters

Type	Name	Description
long	_ySpeed	Speed of dome movement on Y (vertical) axis. Valid range: [-100 .. +100]
long	_camera	ID of camera Valid range: [0 .. 15]
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 if **_xSpeed** or **_ySpeed** parameters are out of range.

-3 if **_camera** is out of range.

-4 if the specified camera is not recognized as a dome.

Remarks

Set **_xSpeed** and **_ySpeed** to 0 to stop the dome.

ZoomDome Method

Purpose

Zoom in/out with the specified camera dome.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long ZoomDome(long _zoom, long _camera, long _connection)

Input Parameters

Table 2-46: Input Parameters

Type	Name	Description
long	_zoom	Speed of zoom. Valid range [-100..100]. A positive value will zoom in and a negative value will zoom out.
long	_camera	ID of camera Valid range: [0 .. 15]
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 if the **_zoom** parameter is out of range.

-3 if **_camera** is out of range.

-4 if the specified camera is not recognized as a dome.

IrisDome Method

Purpose

Sets the iris value on a camera dome.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long IrisDome(long _iris, long _camera, long _connection)

Input Parameters

Table 2-47: Input Parameters

Type	Name	Description
long	_iris	Iris opening value. Valid range [-100..100]. A positive value will open the iris. A negative value will close it.
long	_camera	ID of camera Valid range: [0 .. 15]
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 if the **_iris** parameter is out of range.

-3 if **_camera** is out of range.

-4 if the specified camera is not recognized as a dome.

FocusDome Method

Purpose

Sets the focus value on the camera dome.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long FocusDome(long _focus, long _camera, long _connection)

Input Parameters

Table 2-48: Input Parameters

Type	Name	Description
long	_focus	Focus value. Valid range [-100..100]. A negative value will focus far; a positive value will focus near.
long	_camera	ID of camera Valid range: [0 .. 15]
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 if the **_focus** parameter is out of range.

-3 if **_camera** is out of range.

-4 if the specified camera is not recognized as a dome.

Remarks

Set **_focus** to 0 to stop the dome focus from moving.

SelectPresetDome Method

Purpose

Orders the dome to move to a specified preset.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SelectPresetDome(long _preset, long _camera, long _connection)

Input Parameters

Table 2-49: Input Parameters

Type	Name	Description
long	_preset	Number of the preset to which the dome will move.
long	_camera	ID of camera Valid range: [0 .. 15]
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Return

1 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 if the **_preset** parameter is out of range.

-3 if **_camera** is out of range.

-4 if the specified camera is not recognized as a dome.

SelectTourDome Method

Purpose

Orders the dome to start or stop the specified tour.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SelectTourDome(long _tour, long _camera, long _connection, long _on);

Input Parameters

Table 2-50: Input Parameters

Type	Name	Description
long	_tour	Number of the preset to which the dome camera will move.

Table 2-50: Input Parameters

Type	Name	Description
long	_camera	ID of camera Valid range: [0 .. 15]
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	_on	Boolean. Set to 1 to **start** the tour. Set to 0 to **stop** the tour.

Return

1 on success.

-1 if a generic error occurred or the connection ID is invalid.

-2 if the **_tour** parameter is out of range.

-3 if **_camera** is out of range.

-4 if the specified camera is not recognized as a dome.

SPOT MONITOR METHODS

This group contains all the methods necessary to menage Spot Monitors in normal and custom modes, where supported. These methods are:

- StartCustomSP (see page 2-58)
- StopCustomSP (see page 2-59)
- SetCustomCameraSP (see page 2-60)
- SetCustomDataSP (see page 2-61)
- ResetCustomDataSP (see page 2-62)

StartCustomSP Method

Purpose

Sets the specified monitor to work in custom mode. Nothing will be shown until a *SetCustomCameraSP* or *SetCustomDataSP* method is called.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long StartCustomSP(long connection, long spot_monitor)

Input Parameters

Table 2-51: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	spot_monitor	Number of spot monitors selected.

Return

1 on success.

Remarks

While in custom mode, a single camera can be viewed using *SetCustomCameraSP*, or a cycle can be shown using *SetCustomDataSP*.

StopCustomSP Method

Purpose

Exits the specified monitor from custom mode.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long StartCustomSP(long connection, long spot_monitor)

Input Parameters

Table 2-52: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	spot_monitor	Number of spot monitors selected.

Return

1 on success.

Remarks

Once out of the custom mode, you cannot use *SetCustomCameraSP*, *SetCustomDataSP* and *ResetCustomDataSP*, but you can call *SetSpotMonitor* and *ResetSpotMonitor*.

SetCustomCameraSP Method

Purpose

Sets a camera to be displayed on the specified spot monitor while in custom mode.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long SetCustomCameraSP(long connection, long spot_monitor, long camera, long show_text);

Input Parameters

Table 2-53: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Table 2-53: Input Parameters

Type	Name	Description
long	spot_monitor	Number of spot monitors selected.
long	camera	Number of the camera to be shown. Valid range [0..15]
long	show_text	Boolean: 1 **shows** overlay text on the spot monitor. 0 **does not show** overlay text on the spot monitor.

Return

1 on success.

Remarks

Before using *SetCustomCameraSP*, enter in custom mode by calling *StartCustomSP* method.

SetCustomDataSP Method

Purpose

Sets a camera cycle to be displayed on the specified spot monitor while in custom mode.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long SetCustomDataSP(long connection, long spot_monitor, long cameras, long show_text, long seq_time);

Input Parameters

Table 2-54: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	spot_monitor	Number of spot monitors selected.
long	cameras	A bit-field that determines in the less significant bits which cameras must be displayed.
long	show_text	Boolean: 1 **shows** overlay text on the spot monitor. 0 **does not show** text on the spot monitor.
long	seq_time	Seconds of video of each camera in the cycle.

Return

1 on success.

Remarks

Before using *SetCustomDataSP*, enter in custom mode by calling *StartCustomSP* method.

ResetCustomDataSP Method

Purpose

Resets the custom cycle. Nothing will be shown in the Spot monitor.

Compatibility

DVN 3000: Yes

DVN 5000: No

Syntax

long ResetCustomDataSP(long connection, long spot_monitor)

Input Parameters

Table 2-55: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	spot_monitor	Number of spot monitors to be reset.

Return

1 on success.

Remarks

Before using *ResetCustomDataSP*, enter in custom mode by calling the *StartCustomSP* method.

EXPORT METHODS

This group contains all of the methods necessary to export Audio/Visual streams to AVI or PSV files. These methods are:

- ExportStart (see page 2-63)
- GetExportInfo (see page 2-65)
- ExportStop (see page 2-66)
- ExportMedia (see page 2-67)

ExportStart Method

Purpose

Instructs the server to start sending packets of images from and to specific dates for export purposes.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long ExportStart(long connection, long * startDate, long * endDate, long camera, long sector, long maxBufSize, long maxImages);

Input Parameters

Table 2-56: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long*	startDate	Pointer to an **_int64** containing the starting hour and date of the fragment to export.
long*	endDate	Pointer to an **_int64** containing the ending hour and date of the fragment to export.
long	camera	Number of the camera to export.
long	sector	Number of the sector to export. 0 = Prime 1 = Time Lapse 2 = Alarms
long	maxBufSize	Expressed in bytes – the maximum size of the image buffer returned. 0 = automatic.
long	maxImages	Maximum number of frames to export. Valid range: [1 .. 10] Set this value between 6 and 10 for maximal performance.

Return

0 on success.

-1 if the connection ID is invalid.

Remarks

When this method is called, data will be returned via the *OnExportData* event. When all export data has been sent, or if an error occurs, an *OnExportStatus* event will be triggered.

No more than one export stream can be started for each connection.

See Also

- "GetExportInfo Method" on page 2-65
- "ExportStop Method" on page 2-66
- "OnExportData Event" on page 2-108
- "OnExportStatus Event" on page 2-108

GetExportInfo Method

Purpose

Requests information from the server about the export process.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long GetExportInfo(long connection, long * startDate, long * endDate, long camera, long sector, long infoType);

Input Parameters

Table 2-57: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long*	startDate	Pointer to an **_int64** containing the starting hour and date of the fragment to export.
long*	endDate	Pointer to an **_int64** containing the ending hour and date of the fragment to export.
long	camera	Number of the camera to export.
long	sector	Number of the sector to export. 0 = Prime 1 = Time Lapse 2 = Alarms

Return

a non-negative Stream ID on success.

-1 if the connection ID, the camera, or the sector is invalid.

ExportStop Method

Purpose

Aborts the export process.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long ExportStop(long connection, long streamId);

Input Parameters

Table 2-58: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	streamId	The Stream ID returned by the *ExportStart()* method.

Return

0 on success.

-1 if the connection ID, the camera, or the sector is invalid.

ExportMedia Method

Purpose

Used to export audio/visual streams to .AVI files via a single call to the SpectivaWeb ActiveX control. When called, the ExportMedia method instructs the server to start sending packets of images from and to specific dates for exporting purposes without using pointers as its input parameters.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

long ExportMedia(long connection, long camera, long sector, long codec, long resolution, long splitsize, BSTR aviFile, long startYear, long startMonth, long startDay, long startHour, long startMin, long startSec, long endYear, long endMonth, long endDay, long endHour, long endMin, long endSec, long authenticateMedia);

Input Parameters

Table 2-59: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by the *Connect()* or *CryptConnect()* method.
long	camera	Number of the camera to export. Camera numbering starts at 0 (e.g. use 0 to export from the first camera, 1 to export from the second camera, etc.).
long	sector	Number of the sector to export. 0 = Prime 1 = Time Lapse 2 = Alarm
long	codec	Number of the codec to use. 6 = Multi Codec 7 = Microsoft AVI Codec
long	resolution	Number of the resolution to export. 8 = D1 9 = Half D1 10 = CIF 11 = QCIF

Table 2-59: Input Parameters

Type	Name	Description
long	split_size	Maximum size (in bytes) to export in a file. Use 0 to disable file splitting.
BSTR	aviFile	File path to save.
long	startYear	Year to start export process.
long	startMonth	Month to start export process.
long	startDay	Day to start export process.
long	startHour	Hour to start export process.
long	startMin	Minute to start export process.
long	startSec	Second to start export process.
long	endYear	Year to stop export process.
long	endMonth	Month to stop export process.
long	endDay	Day to stop export process.
long	endHour	Hour to stop export process.
long	endMin	Minute to stop export process.
long	endSec	Second to stop export process.
long	authenticateMedia	1 = Add a digital watermark to the exported file 0 = Do not add the watermark

Return

0 on success.

-1 if there is no available stream or there is a connection error.

Remarks

When this export method is called, the system will report the file completion percentage via the *OnExportMediaStatus* event.

After the system sends all export data, an *OnExportMediaDone* event will be triggered.

If an error occurs, an *OnExportMediaFileError* event will be triggered.

A process which is using the SDK can only have one export stream active at a time.

TEXT INSERTION METHODS

This group contains all of the methods to set up the display of text insertion data and the creation of text insertion data for insertion in the recordings archive. The Text Insertion methods are:

- SetTextInsertionOptions (see page 2-69)
- GetTextInsertionOptions (see page 2-70)
- SendDeviceTextInsertionData (see page 2-72)

SetTextInsertionOptions Method

Purpose

To set up display options for the text insertion feature.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long SetTextInsertionOptions(long connection, long view_live, long view_play, long top, long left, long bottom, long right)

Input Parameters

Table 2-60: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	view_live	Boolean: 1 displays text acquired from a text insertion device in a live window of the specified connection. 0 **does not** display text acquired from a text insertion device in a live window of the specified connection.
long	view_play	Boolean: 1 displays text acquired from a text insertion device on playback windows of the specified connection. 0 **does not** display text acquired from a text insertion device on playback windows of the specified connection.

Table 2-60: Input Parameters

Type	Name	Description
long	top	Upper ordinates of the text display box. Valid Range: [0 .. bottom)
long	left	Left abscissas of the text display box. Valid Range: [0 .. right)
long	bottom	Lower ordinates of the text display box. Valid Range: (top .. 70]
long	right	Right abscissas of the text display box. Valid Range: (left .. 84]

Return

1 on success.

0 if a coordinate is invalid.

-1 if the connection ID is invalid.

Remarks

All coordinates about the text insertion box are referred to a grid of 84x70 cells, regardless of the window resolution.

The setup data is common for every window (both live and playback) of the specified connection.

See Also

"GetTextInsertionOptions Method" on page 2-70

GetTextInsertionOptions Method

Purpose

Call this method to retrieve settings for text insertion for the specified connection.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long GetTextInsertionOptions(long connection, long* view_live, long* view_play, long* top, long* left, long* bottom, long* right

Input Parameters

Table 2-61: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long*	view_live	Boolean: 1 displays text acquired from a text insertion device in a live window of the specified connection. 0 **does not** display text acquired from a text insertion device in a live window of the specified connection.
long*	view_play	Boolean: 1 displays text acquired from a text insertion device on playback windows of the specified connection. 0 **does not** display text acquired from a text insertion device on playback windows of the specified connection.
long*	top	Upper ordinates of the text display box.
long*	left	Left abscissas of the text display box.
long*	bottom	Lower ordinates of the text display box.
long*	right	Right abscissas of the text display box.

Return

1 on success.

Remarks

All coordinates about the text insertion box are referred to a grid of 84x70 cells, regardless of the window resolution.

The setup data is common for every window (both live and playback) of the specified connection.

See Also

“SetTextInsertionOptions Method” on page 2-69

SendDeviceTextInsertionData Method

Purpose

Use this method to insert text insertion data in the storage.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

SendDeviceTextInsertionData(long connection, long id, long cameras, BSTR text)

Input Parameters

Table 2-62: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	id	ID of the device that generates the text. This ID must be a valid text insertion device ID created in the server setup and marked as **Ethernet device**.
long	cameras	A bit-field that determines the camera for which data will be stored. Use only the 16 less significant bits: bit 1 for camera 1, bit 2 for camera 2, and so on.
BSTR	text	Text to be inserted in the archive (and/or shown in the text box).

Return

1 on success.

Remarks

Maximum text size is 200 characters. If zero is sent as the value for the cameras parameter, then the cameras associated with the text insertion device in the DVN's text insertion configuration will be used instead. This method is recommended for high throughput applications, where the overhead associated with decoding and processing the cameras parameter can result in lost text insertion displays for live viewing.

MANUAL RECORDING METHODS

This group contains all methods to handle manual recording. These are:

- StartManualRec (see page 2-73)
- StopManualRec (see page 2-74)
- ResetManualRec (see page 2-75)
- GetRecordingMode (see page 2-76)

StartManualRec Method

Purpose

Starts the manual recording of the specified camera on the specified sector.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long StartManualRec(long connection, long camera, long sector)

Input Parameters

Table 2-63: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	camera	Number of the camera from which to manually start recording.
long	sector	Number of the sector on which the recorded video data will reside.

Return

1 on success.

Remarks

Use the following constants values for the sector parameter:

Table 2-64: Sector Values

Sector	Value
PRIME	0
TIME LAPSE	1
ALARMS	2

StopManualRec Method

Purpose

Stops the manual recording of the specified camera on the specified sector, regardless of its standard scheduler.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long StopManualRec(long connection, long camera, long sector)

Input Parameters

Table 2-65: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	camera	Number of the camera from which to manually stop recording.
long	sector	Number of the sector.

Return

1 on success.

Remarks

Use following constants values for the sector parameter:

Table 2-66: Sector Values

Sector	Value
PRIME	0
TIME LAPSE	1
ALARMS	2

ResetManualRec Method

Purpose

Stops the manual recording of the specified camera on the specified sector; but the system switches back to its standard scheduler.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long StopManualRec(long connection, long camera, long sector)

Input Parameters

Table 2-67: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	camera	Number of the camera to remove from the forced recording on the specified sector.
long	sector	Number of the sector to remove from the forced recording.

Return

1 on success.

Remarks

Use following constants values for the sector parameter:

Table 2-68: Sector Values

Sector	Value
PRIME	0
TIME LAPSE	1
ALARMS	2

GetRecordingMode Method

Purpose

Retrieves from the server the status of the recording on the specified camera and sector. An *OnRecordingMode* event is triggered when the status is received.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

long GetRecordingMode(long connection, long camera, long sector)

Input Parameters

Table 2-69: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	camera	Number of the camera to remove from the forced recording on the specified sector.
long	sector	Number of the sector to remove from the forced recording.

Return

1 on success.

Remarks

The return value **is not** the status of the recording. This is provided by an *OnRecordingMode* event. Use following constants values for the sector parameter:

Table 2-70: Sector Values

Sector	Value
PRIME	0
TIME LAPSE	1
ALARMS	2

See Also

"OnRecordingMode Event" on page 2-113

EVENTS

The following events are provided by the event dispatch interface of the ActiveX control:

Table 2-71: Event Descriptions

Event	Description	See Page . . .
WatchDog	Triggered when the watchdog message is received from the DVR server.	2-79
SetupError	The setup received was corrupted or not compatible with the client.	2-80
OnVideoError	An error occurred when attempting to display video.	2-80
BlockingEvent	Entering or exiting in a blocking operation.	2-81
PauseEvent	DEPRECATED	---
CamerasReceived	After the ActiveX received the setup structure, this event is triggered to pass general information about all connected cameras.	2-82
OnCreate	The first event triggered as the ActiveX object is created.	2-83
OnDoubleClick	The user has double-clicked on a camera window.	2-84
OnConnection	Replay for a connection request.	2-84
OnAlarm	The server sends a notification when an alarm has changed.	2-85
OnAlarmAvailableEx	Returns the available alarms, without the limit to 16 alarms. (USB Extension board support present).	2-86
OnAux	Server sends notification when an AUX has changed.	2-87

Table 2-71: Event Descriptions

Event	Description	See Page . . .
OnAuxAlarmStatus	Status of all alarms and AUXes sent to client.	2-88
OnCameraChanged	In mono mode, displays an image of a camera different from the actual one.	2-89
OnBuffering	The first fragment of the first image of a stream is received, but still waiting for the other “chunks.”	2-90
OnDomesStatus	Notifies whether camera is standard or dome type.	2-90
OnPlayStatus	After setup is received, notifies if playback is possible.	2-91
OnRecording	Recording of the specified camera has started or stopped.	2-92
OnDarkening	Darkening detected on the specified camera.	2-93
OnDateSync	Sends notification about the exact hour and date of the server.	2-94
OnLoadingLocalPlay	DEPRECATED	---
AudioChannelsOn	Notifies which audio channel(s) is associated with a camera.	2-94
OnAuxAlarmStatusEx	Notifies the status of AUXes and alarms (but USB extension board is not supported).	2-95
OnAlarmsAvailable	Returns the availability of the alarms connector, both physical or logical.	2-96
OnAuxAvailable	Returns the availability of the AUX connector (but USB extension board is not supported).	2-97
OnCamerasDisconnected	Notifies which camera is active, but disconnected.	2-98
OnCamerasTooLarge	Notifies which camera is active, but the image is too large to be recorded or viewed.	2-99
OnFindData	Returns a buffer of A/V stream fragments available.	2-100
OnStreamInfo	Periodically sends date and hour of the last image received.	2-101
OnAuxStatus	Returns the status of up to 128 AUXes. USB extension board is supported.	2-102
OnAlarmStatus	Returns the status of up to 128 alarms. USB extension board is supported.	2-103
OnAuxAvailableEx	Returns the availability of up to 128 AUX connectors. USB extension board is supported.	2-104
OnStreamImage	Internal Use Only	---
OnChannelsAudioAvailable	Notifies which audio channels are available on the server.	2-105
OnSectorsAvailable (DVN3000)	When the ActiveX receives the setup structure, this event is triggered and provides general information about all connected cameras.	2-106
OnSectorsAvailable (DVN5000)	Returns which camera is enabled to record on which sector.	2-106

Table 2-71: Event Descriptions

Event	Description	See Page . . .
OnWinStatusChange	Notifies why a window has passed from playback to live, or from live to playback.	2-107
OnExportInfo	Internal Use Only	---
OnExportData	Returns data required to export.	2-108
OnExportStatus	Returns the status of the export process.	2-108
OnExportMediaStatus	Returns the status of the *ExportMedia* method.	2-109
OnExportMediaDone	Triggered when the export process of the *ExportMedia* method is completed.	2-110
OnExportMediaFileError	Triggered if an error occurs during the *ExportMedia* method export process.	2-110
OnCameraAuxAvailable	Returns a bit field of available AUXes for each camera.	2-111
OnCameraAuxStatus	Returns a bit field of the status of the AUXes of each camera.	2-111
OnCameraAux	Returns the status of a changed AUX of a camera.	2-112
OnRecordingMode	Returns the recording status of the specified camera and sector.	2-113
OnAlarmEx	Returns extended information about an alarm.	2-114
OnNeedCertificate	When certification is needed, data verification is requested.	2-115

WatchDog Event

Purpose

Sends notification when the watchdog message is received from the server (the connection is still active).

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

WatchDog(long _connection)

Input Parameters

Table 2-72: Input Parameters

Type	Name	Description
long	_connection	ID of server connection for which the watchdog message is sent. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

SetupError Event

Purpose

The setup received was corrupted or (more probably) the setup version is not compatible with the client.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

SetupError (long _connection)

Input Parameters

Table 2-73: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnVideoError Event

Purpose

Sends notification when an error occurs when attempting to display video.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnVideoError (long _error)

Input Parameters

Table 2-74: Input Parameters

Type	Name	Description
long	_error	Error code: 0 = Error initializing graphic engine. Common when using DVN3000Web ActiveX Version 3.0.0.26 with a 16-bit color depth display. 1 = Other error.

BlockingEvent Event

Purpose

Sends notification when the ActiveX control is entering or exiting in a blocking operation. While in a blocking operation, other events cannot be triggered and other methods cannot be performed.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

BlockingEvent(long _blocking)

Input Parameters

Table 2-75: Input Parameters

Type	Name	Description
long	_blocking	Block Status: 1 = Actually entering into the blocking code fragment. 0 = The blocking operation has just terminated.

Remarks

While in a blocking operation, no other event will be triggered until another *BlockingEvent(0)* is sent, indicating that the blocking operation has finished.

If a method has been called in the mean time, it will be correctly executed.

CamerasReceived Event

Purpose

Notifies when the ActiveX receives a new setup file from the server. Then a string is passed containing basic data about the server name and the camera name.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

CamerasReceived(BSTR _names, long _position)

Input Parameters

Table 2-76: Input Parameters

Type	Name	Description
BSTR	_names	A formatted string containing server and camera names. See “Remarks” on page 2-83 for details.
long	_position	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

_names is a string formatted in the following manner:

```
Server name
Marker (*)
Camera 1 name.
Marker (*)
Camera 2 name.
Marker (*)
Camera 3 name.
Marker (*)
Camera 4 name.
Marker (*)
...
Camera 16 name.
Marker (*)
```

If the first character of a camera name is an exclamation mark (!), the camera is disconnected.

OnCreate Event

Purpose

This event is triggered as soon as the ActiveX control is created. This event is always triggered first.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCreate()

Remarks

When this event is received, methods can be called; otherwise, they would be refused.

OnDoubleClick Event

Purpose

This event indicates that the local user has double-clicked a window in which the ActiveX was drawing an A/V stream.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnDoubleClick(long _window)

Input Parameters

Table 2-77: Input Parameters

Type	Name	Description
long	_window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.

OnConnection Event

Purpose

Notification of an accepted or rejected connection request to the server, including the reason for the rejection (see “Remarks” on page 2-85).

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnConnection(long _status, long _connection)

Input Parameters

Table 2-78: Input Parameters

Type	Name	Description
long	_status	Connection error code. See “Remarks” on page 2-85 for code details.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

OnConnection(long _status, long _connection)

Table 2-79: _status Values and Descriptions

_status Value	Description
Less than 0	Windows networking error. If the error code is less than 0, you must change it to a positive number. To find a description of the error code, or to change the code, locate and change it using the Windows system. Refer to the Windows documentation for assistance.
0	Disconnected by server.
1	Valid connection.
2	Unable to find server.
3	Wrong user name or password.
4	Wrong or missing license.
5	Black-listed client.
6	Wrong setup version.
7	Too many users connected.

OnAlarm Event

Purpose

Notifies the status of an alarm changed on the server.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnAlarm(long _numAlarm, long _status, long _connection)

Input Parameters

Table 2-80: Input Parameters

Type	Name	Description
long	_numAlarm	Number of the changed alarm.
long	_status	Boolean: 1 = alarm **active**. 0 = alarm **not active**.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnAlarmAvailableEx Event

Purpose

Returns the availability of alarms in the given server. This event is triggered as the connection is established and every time the setup is modified. USB Extension board is supported.

Compatibility

DVN 3000: Yes, since release 3.05.00

DVN 5000: No

Syntax

OnAlarmAvailableEx(long* _status, long _size, long _connection)

Input Parameters

Table 2-81: Input Parameters

Type	Name	Description
long*	_status	Pointer to bit field of **_size** bytes, organized into DWORD (32 bits). _status[0] contains availability of alarms from 0 to 31, _status[1] contains availability of alarms from 32 to 63, and so on. 1 = available 0 = not available
long	_size	Size in bytes of the data structure pointed at **_status**.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

NOTE
*In DVN3000 Version 3.05.00, one only DWORD is returned, as 32 alarms are supported, so **_size** contains 4.*

OnAux Event

Purpose

Server sends notification when an AUX has changed.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnAux(long _numAlarm, long _status, long _connection)

Input Parameters

Table 2-82: Input Parameters

Type	Name	Description
long	_numAux	Number of the changed AUX.

Table 2-82: Input Parameters

Type	Name	Description
long	_status	Boolean: 1 = AUX **active** 0 = AUX **not active**
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnAuxAlarmStatus Event

Purpose

The server notifies the client the status of **all** DVN 3000 alarms and AUXes. If this event is triggered in a DVN 5000 ActiveX control, only data relative to the first 16 AUXes and Alarms is provided.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnAuxAlarmStatus(long _status, long _connection)

Input Parameters

Table 2-83: Input Parameters

Type	Name	Description
long	_status	A bit-field containing the status of all alarms and AUXes. See “Remarks” on page 2-88 for details.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

_status parameter structure: the 32 bit parameter is a bit-field in which the lower 16 bits represent the activation status of alarms (1 for active, 0 for inactive), while the higher 16 bits represent the activation status of AUXes (1 for active, 0 for inactive).

See Also

- "OnAuxStatus Event" on page 2-102
- "OnAlarmStatus Event" on page 2-103

OnCameraChanged Event

Purpose

When viewing streams in mono mode, an image is received that belongs to a camera different from the actual one.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCameraChanged(long _camera, long _connection)

Input Parameters

Table 2-84: Input Parameters

Type	Name	Description
long	_camera	Number of the camera that will be displayed from now on.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

This event is triggered only when using one bare window for displaying A/V streams and the image received belongs to a camera different from the one actually displayed. As a consequence of the *SetCamera()* method call, the first image of the new camera has been received.

OnBuffering Event

Purpose

The first fragment of the first image of a stream is received, but you are still waiting for the other “chunks.”

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnBuffering(long _value, long _connection)

Input Parameters

Table 2-85: Input Parameters

Type	Name	Description
long	_value	Boolean: 1 = buffering **beginning** 0 = buffering **terminated**
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

This event is useful with slow connections (56 Kb or slower) where the waiting time before receiving a complete first frame could last a few seconds. For example, if the connection speed is single line ISDN (64Kbps) and the image quality is high, and no delta transmission is active, a single full frame could take more than 10 seconds. This event can be used to inform the user that the connection functions correctly and the image is being received, but he/she must wait a moment longer.

OnDomesStatus Event

Purpose

Notifies the client whether the camera is configured as a dome or a standard camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnDomesStatus(long _cameras, long _connection)

Input Parameters

Table 2-86: Input Parameters

Type	Name	Description
long	_cameras	A bit-field containing a flag in the lower 16 bits indicating the dome presence for each camera. 1 = camera **is** a dome 0 = camera **is not** a dome
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnPlayStatus Event

Purpose

After the setup is received, this event notifies the client whether playback is possible.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnPlayStatus(long _status, long _connection)

Remarks

The playback may not be available if no recorded material is present or if the connected user is not allowed to view it.

Input Parameters

Table 2-87: Input Parameters

Type	Name	Description
long	_status	Boolean: 1 = playback **enabled** 0 = playback **disabled**
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnRecording Event

Purpose

Sends notification when the recording of the specified camera has started or stopped.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnRecording(long _camera, long _status, long _connection)

Input Parameters

Table 2-88: Input Parameters

Type	Name	Description
long	_camera	Number of the camera of which has changed the recording status.
long	_status	Boolean: 1 if the recording has **started**. 0 if the recording has **stopped**.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

This event is triggered even if the specified camera is not viewed. After this event, an *OnDarkening* event is always triggered.

OnDarkening Event

Purpose

Sends notification when the darkening on the specified camera is detected or is no longer detected.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnDarkening(long _camera, long _status, long _connection)

Input Parameters

Table 2-89: Input Parameters

Type	Name	Description
long	_camera	Number of the camera of which has changed the darkening status.
long	_status	Boolean: 1 if the darkening is detected. 0 if the darkening is no longer detected.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

This event is triggered even if the specified camera is not viewed. Before this event, an *OnRecording* event is always triggered.

OnDateSync Event

Purpose

Notifies the client of the exact hour and date of the server.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnDateSync(long _hour, long _min, long _sec, long _day, long _month, long _year, long _connection)

Input Parameters

Table 2-90: Input Parameters

Type	Name	Description
long	_hour	Hour of the server
long	_min	Minutes of the server
long	_sec	Seconds of the server
long	_day	Day of the server
long	_month	Month of the server
long	_year	Year of the server
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

No information is provided about time zone or daylight saving time. Hour and date are provided as shown on the server.

AudioChannelsOn Event

Purpose

Notifies which audio channels are associated with the given camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

AudioChannelsOn(long _connection, long _channels, long _camera)

Input Parameters

Table 2-91: Input Parameters

Type	Name	Description
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	_channels	A 32-bit bit-field. The less significant bit is channel 1; the most significant is channel 32. 1 = This channel **is** associated with the camera. 0 = This channel **is not** associated with the camera. The DVN 5000 supports up to 4 channels associated with a camera, while the DVN 3000 supports only 1 channel.
long	_camera	Number of the camera.

OnAuxAlarmStatusEx Event

Purpose

Returns the status for 16 AUXes and 128 alarms.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnAuxAlarmStatusEx(long * _status, long _connection);

Input Parameters

Table 2-92: Input Parameters

Type	Name	Description
long*	_status	Pointer to a 5 DWORD array containing a bit-field of information about AUX and alarm status. See “Remarks” on page 2-96 for details.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

Build the 5 DWORD array in the following manner:

```
DWORD 0: 0..15 = auxes 0..15
DWORD 1: 0..31 = alarms  0..31
DWORD 2: 0..31 = alarms 32..63
DWORD 3: 0..31 = alarms 64..95
DWORD 4: 0..31 = alarms 96..127
```

Bit set (1) means state is ON.

Bit clear (0) means state is OFF.

In this case, the number of AUXes is limited to 16, so the USB extension board in installed AUXes 16 to 127 will not be received.

This event will be dismissed, so use *OnAlarmStatus* and *OnAuxStatus* instead.

See Also

- “OnAlarmStatus Event” on page 2-103
- “OnAuxStatus Event” on page 2-102

OnAlarmsAvailable Event

Purpose

Returns the availability of the 128 alarms.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnAlarmsAvailable(long * _status, long _connection)

Input Parameters

Table 2-93: Input Parameters

Type	Name	Description
long*	_status	Pointer to a 4 DWORD array containing a bit-field of information about alarm availability. See “Remarks” on page 2-97 for details.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

Build the 4 DWORD array in the following manner:

```
DWORD 0: 0..31 = alarms  0..31
DWORD 1: 0..31 = alarms 32..63
DWORD 2: 0..31 = alarms 64..95
DWORD 3: 0..31 = alarms 96..127
```

Bit set (1) means state is ON.

Bit clear (0) means state is OFF.

OnAuxAvailable Event

Purpose

Returns the availability of the 128 AUX connectors.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnAuxAvailable(long _status, long _connection)

Input Parameters

Table 2-94: Input Parameters

Type	Name	Description
long*	_status	Pointer to a 4 DWORD array containing a bit-field of information about AUX availability. See “Remarks” on page 2-98 for details.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

Build the 4 DWORD array in the following manner:

```
DWORD 0: 0..31 = AUXes  0..31
DWORD 1: 0..31 = AUXes 32..63
DWORD 2: 0..31 = AUXes 64..95
DWORD 3: 0..31 = AUXes 96..127
```

Bit set (1) means state is ON.

Bit clear (0) means state is OFF.

OnCamerasDisconnected Event

Purpose

Sends notification when cameras set as active are disconnected.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCamerasDisconnected(long _status, long _connection)

Input Parameters

Table 2-95: Input Parameters

Type	Name	Description
long	_status	A bit-field containing in the 16 less significant bits which camera is active but disconnected. Bit 0 = camera 1 Bit 15 = camera 16
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnCamerasTooLarge Event

Purpose

Sends notification when a camera is not being viewed (nor recorded) because the image is too large (exceeding 100 KB per frame).

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCamerasTooLarge(long _status, long _connection)

Input Parameters

Table 2-96: Input Parameters

Type	Name	Description
long	_status	A bit-field containing in the 16 less significant bits which camera is in "Too Large" status. Bit 0 = camera 1 Bit 15 = camera 16
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnFindData Event

Purpose

As a request of a *GetFindData(),* an *OnFindData* event is triggered containing information about the recorded material. See “Remarks” on page 2-100 for further details.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnFindData(long connection, long camera, long* info, long countIntervals, long* data, long finished)

Input Parameters

Table 2-97: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	camera	Camera for which the find data is provided.
long*	info	Pointer to a **findDataStruc** record a containing previously submitted find request.
long	countIntervals	Number of intervals returned.
long*	data	Pointer to [countIntervals] pairs of _int64 representing each interval beginning and finish time.
long	finished	Unused.

Remarks

The FIND operation begins with call to a *GetFindData* method, where the last parameter is a pointer to a **FindDataStruc** record (see “GetFindData Method” on page 2-40 for details), containing a generic time interval and a SendData value of 2 (initial request).

The answer to this request is an *OnFindData* event where the field information contains a pointer to another **FindDataStruc** record with the precise beginning and ending hour, the filter applied, and a sendData value indicating find request in progress.

This record returned is used to perform a second *GetFindData* request. At this point, an *OnFindData* event will be triggered, where the parameter **countIntervals** contains the real number of intervals returned. The parameter **data** contains a pointer to an array of [countIntervals] records composed by 2 _int64, where the first one is the beginning hour of an interval and the second one is the ending time of that same interval. The **info** field contains a pointer to a **FindDataStruc** record which should be equal to the other **FindDataStruc** previously returned. In case of difference, the fist received is the valid one.

See Also

"GetFindData Method" on page 2-40

OnStreamInfo Event

Purpose

Periodically notifies the client the date and hour when the last image was received and displayed.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnStreamInfo(long window, long* date, long status)

Input Parameters

Table 2-98: Input Parameters

Type	Name	Description
long	window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long*	date	Pointer to an _int64 containing the date and hour of the last image viewed in the window specified in the field **window**.
long	status	Unused

Remarks

The period of this event can be modified or disabled by calling the *RequestStreamInfo* method.

See Also

RequestStreamInfo() Method

OnAuxStatus Event

Purpose

Returns the status of the AUXes.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnAuxStatus(long * _status, long _size, long _connection)

Input Parameters

Table 2-99: Input Parameters

Type	Name	Description
long*	_status	Pointer to a (_size/4) DWORD array containing a bit-field of information about the AUXes' availability. See "Remarks" on page 2-102 for details.
long	_size	Size in bytes of the array pointed at **_status**.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

As this event supports virtually infinite AUXes, know the size of the data returned. This size is present in the **_size** parameter and is expressed in bytes, so the number of DWORD returned is **_size/4**. The number of DWORD returned is provided by the number of AUXes present, divided by 32, then rounded up.

Build the n DWORD array in the following manner:

```
DWORD 0: 0..31 = AUXes  0 .. 31
DWORD 1: 0..31 = AUXes  32 .. 63
...
DWORD N: 0..31 = AUXes  N*32 .. (N+1)*32-1
```

OnAlarmStatus Event

Purpose

Returns the status of the alarms.

Compatibility

DVN 3000: Since Release 3.05.00

DVN 5000: Yes

Syntax

OnAlarmStatus (long * _status, long _size, long _connection)

Input Parameters

Table 2-100: Input Parameters

Type	Name	Description
long*	_status	Pointer to a (_size/4) DWORD array containing a bit-field of information about Alarm availability. See “Remarks” on page 2-103 for details.
long	_size	Size in bytes of the array pointed at **_status**.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

As this event supports virtually infinite alarms, know the size of the data returned. This size is present in the **_size** parameter and is expressed in bytes, so the number of DWORD returned is **_size/4**. The number of DWORD returned is provided by the number of alarms present, divided by 32, then rounded up.

Build the n DWORD array in the following manner:

```
DWORD 0: 0..31 = alarms  0 .. 31
DWORD 1: 0..31 = alarms  32 .. 63
...
DWORD N: 0..31 = alarms  N*32 .. (N+1)*32-1
```

1 = turned on; 0 = turned off.

OnAuxAvailableEX Event

Purpose

Returns the availability of the AUXes; USB extension board is supported.

Compatibility

DVN 3000: Since Release 3.05.00

DVN 5000: Yes

Syntax

OnAuxAvailableEx(long * _status, long _size, long _connection)

Input Parameters

Table 2-101: Input Parameters

Type	Name	Description
long*	_status	Pointer to bit field of **_size** bytes, organized into DWORD (32 bits). _status[0] contains availability of AUXes from 0 to 31, _status[1] contains availability of AUXes from 32 to 63, and so on. 1 = available; 0 = not available
long	_size	Size in bytes of the data structure pointed at **_status**.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

Build the 4 DWORD array in the following manner:

```
DWORD 0: 0..31 = AUXes  0..31
DWORD 1: 0..31 = AUXes 32..63
```

```
DWORD 2: 0..31 = AUXes 64..95
DWORD 3: 0..31 = AUXes 96..127
```

Bit set (1) means state is ON.

Bit clear (0) means state is OFF.

OnChannelsAudioAvailable Event

Purpose

Notifies when the ActiveX has received a new setup file from the server, and specifies which audio channels are available.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnChannelsAudioAvailable(long _status, long _connection)

Input Parameters

Table 2-102: Input Parameters

Type	Name	Description
long	status	A 32-bit bit field. **DVN3000:** Only the 16 less significant bits are used. The less significant bit is channel 1; the most significant bit is channel 16. 1 = this channel **exists**. 0 = this channel **does not** exist. **DVN5000:** All 32 bits are used. The less significant bit is channel 1; the most significant is channel 32. 1 = this channel **exists**. 0 = this channel **does not** exist.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

See Also

"AudioChannelsOn Event" on page 2-94

OnSectorsAvailable Event

Purpose

Specifies which sectors are available for playback for each camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnSectorsAvailable (long * _status, long _connection)

Input Parameters

Table 2-103: Input Parameters

Type	Name	Description
long	status	A pointer to a *cameraSectorStatus*. See "Remarks" on page 2-106 for details.
long	_connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

Every time an image is received, this data is checked. As a difference occurs, this event is launched.

```
struct cameraSectorStatus
{
DWORD Prime;
DWORD TLapse;
DWORD Alarm;
};
```

Each DWORD contains playback availability for each camera. Only the 16 less significant bits in each camera are used.

Bit [0..15] equals camera [1..16]

Values:

0 = sector **unavailable** for playback

1 = sector **available** for playback

OnWinStatusChange Event

Purpose

Notifies the client that it cannot continue playback of the actual sector, so playback will be switched to another sector.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnWinStatusChange(long _window, long _status, long _reason)

Input Parameters

Table 2-104: Input Parameters

Type	Name	Description
long	window	ID of a video display window. This must be a valid window ID returned by *SetExternalWindow()* or *CreateWin()* methods.
long	_status	The sector to which playback will switch. See “Remarks” on page 2-107 for details.
long	_reason	Error code of the error that occurred. See “Remarks” on page 2-107 for details.

Remarks

_status Constants:

```
STATUS_LIVE      = 0   // Switching to Live.
STATUS_PLAY      = 1   // Switching to Playback.
STATUS_PRIME     = 2   // Switching to Prime sector.
STATUS_TLAPSE    = 3   // Switching to Time Lapse sector.
STATUS_ALARM     = 4   // Switching to Alarm sector.
MASTER_WIN       = 5   // [DVN5000 only] Master camera in multiple
                       // playback removed.
```

_reason Constants:

```
NODATA_AVAIL      = 0   // No more playback data available.
PLAY_MULT         = 1   // [DVN5000 only] Switching to multiple playback.
PLAY_MULT_STOPPED = 2   // [DVN5000 only] Multiple playback stopped.
MASTER_REMOVED    = 3   // [DVN5000 only] Master camera playback windows
                        // closed while in multiple playback.
```

OnExportData Event

Purpose

As a consequence of a call to the *ExportStart()* method, this event is triggered when a block of data is sent.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnExportData(long connection, long streamId, long* buffer, long bufferSize)

Input Parameters

Table 2-105: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long*	streamId	A valid StreamID returned by the *ExportStart()* method.
long	buffer	Pointer to a buffer containing data to be processed for export purposes.
long	bufferSize	Size of the buffer.

OnExportStatus Event

Purpose

Status notification of the export process.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnExportStatus(long connection, long streamId, long status)

Input Parameters

Table 2-106: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	streamId	A valid StreamID returned by the *ExportStart()* method.
long	status	Status of the export process: 0 = Export finished 1 = Export error

OnExportMediaStatus Event

Purpose

Returns the status of the ExportMedia method.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnExportMediaStatus(long percentage);

Input Parameters

Table 2-107: Input Parameters

Type	Name	Description
long	percentage	Percentage of file exported during the export process.

OnExportMediaDone Event

Purpose

Triggered when the export process of the *ExportMedia* method is completed.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnExportMediaDone();

OnExportMediaFileError Event

Purpose

Triggered if an error occurs during the *ExportMedia* method export process.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnExportMediaFileError();

OnCameraAuxAvailable Event

Purpose

Notifies the client on the availability of AUX connectors belonging to each camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCameraAuxAvailable(long * status, long size, long connection)

Input Parameters

Table 2-108: Input Parameters

Type	Name	Description
long*	status	An array of size/4 DWORD, containing in each DWORD the availability of AUXes belonging to each camera. See “Remarks” on page 2-111 for details.
long	size	Size in bytes of the data structure pointed at ***status**.
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

Up to 32 AUXes are supported for each camera.

DWORD 0 contains AUX availability for camera 0, DWORD 1 contains AUX availability for camera 1, and so on.

In each DWORD:

bit 0: 1 = AUX 0 available; 0 = AUX 0 unavailable

bit 31: 1 = AUX 31 available; 0 = AUX 31 unavailable.

OnCameraAuxStatus Event

Purpose

Notifies the client the status of AUX connectors belonging to each camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCameraAuxStatus(long * status, long size, long connection)

Input Parameters

Table 2-109: Input Parameters

Type	Name	Description
long*	status	An array of size/4 DWORD, containing in each DWORD the status of AUXes belonging to each camera. See “Remarks” on page 2-112 for details.
long	size	Size in bytes of the data structure pointed at ***status**.
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

Remarks

DWORD 0 contains AUX status for camera 0; DWORD 1 contains AUX status for camera 1, and so on.

In each DWORD:

bit 0: 1 = AUX 0 closed; 0 = AUX 0 open;

bit 31: 1 = AUX 31 closed, 0 = AUX 31 open.

OnCameraAux Event

Purpose

Notifies the client the status of the specified AUX connector belonging to the specified camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnCameraAux(long camera, long aux, long status, long connection)

Input Parameters

Table 2-110: Input Parameters

Type	Name	Description
long	camera	Number of the camera whose AUX has been modified.
long	aux	Number of the modified AUX.
long	status	New status of the modified AUX: 1 = closed 0 = open
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnRecordingMode Event

Purpose

Notifies the client of the recording status of the specified camera on the specified camera.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnRecordingMode(long camera, long sector, long manualRec, long genericRec, long connection)

Input Parameters

Table 2-111: Input Parameters

Type	Name	Description
long	camera	Number of the camera for which details are provided.
long	sector	Number of the sector for which details are provided.

Table 2-111: Input Parameters

Type	Name	Description
long	manualRec	Real mode of the recording: 0 automatic (normal state after a *ResetManualRec* method call) 1 manual recording (after a *StartManualRec* method call) 2 manual stop (after a *StopManualRec* method call)
long	GenericRec	Boolean: 1 if the sector is in recording state 0 otherwise
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.

OnAlarmEx Event

Purpose

Returns extended information on a triggered alarm.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnAlarmEx (long connection, long alarmId, long alarmType, BSTR alarmDescription, BSTR conditionsDescription, BSTR conditionsStatus, long alarmStatus, long* serverDate)

Input Parameters

Table 2-112: Input Parameters

Type	Name	Description
long	connection	ID of server connection. This must be a valid server connection ID returned by *Connect()* or *CryptConnect()* method.
long	alarmId	Number of the alarm.
long	AlarmType	Type of triggered alarm. See “Remarks” on page 2-115 for details (DVN5000 only).
BSTR	AlarmDescription	Textual description of the alarm.

Table 2-112: Input Parameters

Type	Name	Description
BSTR	conditionsDescription	Full textual description of a custom condition (DVN5000 only).
BSTR	conditionsStatus	Logical status of each sub-condition in the custom condition. (DVN5000 only).
long	alarmStatus	Boolean: 1 = alarm active 0 = alarm not active
long*	serverDate	Pointer to an **_int64** containing the server hour and date during which the alarm was triggered.

Remarks

AlarmType Constants values (DVN5000 only)

ALARMGROUP_NONE -1
ALARMGROUP_MOTION 0
ALARMGROUP_VIDEOLOSS 1
ALARMGROUP_SWITCH 2
ALARMGROUP_IMAGEBEHAVIOUR 3

OnNeedCertificate Event

Purpose

Passes to the client an encrypted buffer to be decrypted and sent back.

Compatibility

DVN 3000: Yes

DVN 5000: Yes

Syntax

OnNeedCertificate(long * buffer, long size, long connection)

Input Parameters

Table 2-113: Input Parameters

Type	Name	Description
long*	buffer	A buffer containing encrypted data to be decrypted and sent back to the server as user validation.

Table 2-113: Input Parameters

Type	Name	Description
long	size	Size in bytes of the data structure pointed at ***buffer**.
long	connection	ID of server connection.

Remarks

The buffer contains encrypted data. This data has to be decrypted using the appropriate certificate. Then, this decrypted data must be sent back to the server by calling the *SendCertificate* method to continue the login process.

See Also

"SendCertificate Method" on page 2-11

OnTIData Event

Purpose

Provides a way for clients to capture Text Insertion (TI) overlay text separately from the video stream, so that the text can be displayed in a separate area of the UI and parsed, if desired, to decide whether or not to display it.

Compatibility

DVN 3000: No

DVN 5000: Yes

Syntax

OnTIData(BSTR title, BSTR prefix, BSTR text, long connection, long device, long cameras)

Input Parameters

Table 2-114: Input Parameters

Type	Name	Description
BSTR	title	The text displayed above the TI text when TI is displayed as video overlay. Configured server-side and applies to all clients.

Table 2-114: Input Parameters

Type	Name	Description
BSTR	prefix	The text displayed to the left of each line of TI text when TI is displayed as video overlay. Configured server-side and applies to all clients.
BSTR	text	The actual TI text – up to 70 characters, one line only.
long	connection	The number of the connection with which the text is associated, as returned by a previous call to *Connect()*.
long	device	The number of the TI Device which generated the text. The number will be between 1 and 16. TI Devices are configured server-side and can represent text coming from either serial or network devices. The text from TI Devices can be associated with one or more cameras in the DVN server configuration.
long	cameras	A bitmask field indicating with which cameras the TI text is associated. For instance, a value of 3 indicates that the TI text is associated with cameras 1 and 2, while a value of 6 indicates the TI text is associated with cameras 2 and 3.

Remarks

Depending on the system configuration, the DVN may capture overlay text from serial or network devices in real time and associate it with video from one or more cameras. When clients view video, they can optionally specify that any overlay text associated with the video should also be shown as overlay in a configurable area within the video display region. However, if clients wish to display the text outside of the video region, or capture it to a file or database, they must use the *OnTIData* event to receive the text separately from the video.

NOTE

*Even if text overlay display has been turned off in the client via a call to **SetTextInsertionOptions()**, TI text can still be received using the **OnTIData** event.*

Using the *OnTIData* event, clients can also implement a filtering capability. For example, if the first 5 characters of each line of TI text contain a numeric transaction typecode, the client UI could have check boxes allowing the user to toggle display/suppression of various typecodes, and the event handler for *OnTIData* could examine the first 5 characters of the text before deciding to display it.

USING A PORTAL PAGE TO PLAY DVN VIDEO CLIPS

This section describes how browser-based applications can use the SDK to play video from a specified DVN 5000 or DVN 3000 on any client computer using a series of calls to the **SpectivaWeb.ocx** ActiveX control from Javascript executing in the application's web page. Any web server and server OS can be used to support the application. With the necessary software components in place, users can play video:

- from a specific DVN camera
- from a specified date
- for a specified number of seconds

The required software components which must be installed on the web/portal server consist of the **SpectivaVisionWEB.cab** file (available as part of the DVN SDK installation), an HTML file (.html), and a Javascript file (.js), all of which are required to view and play the video (see the HTML and Javascript code from the sample files on page 2-120 and 2-121 for guidance).

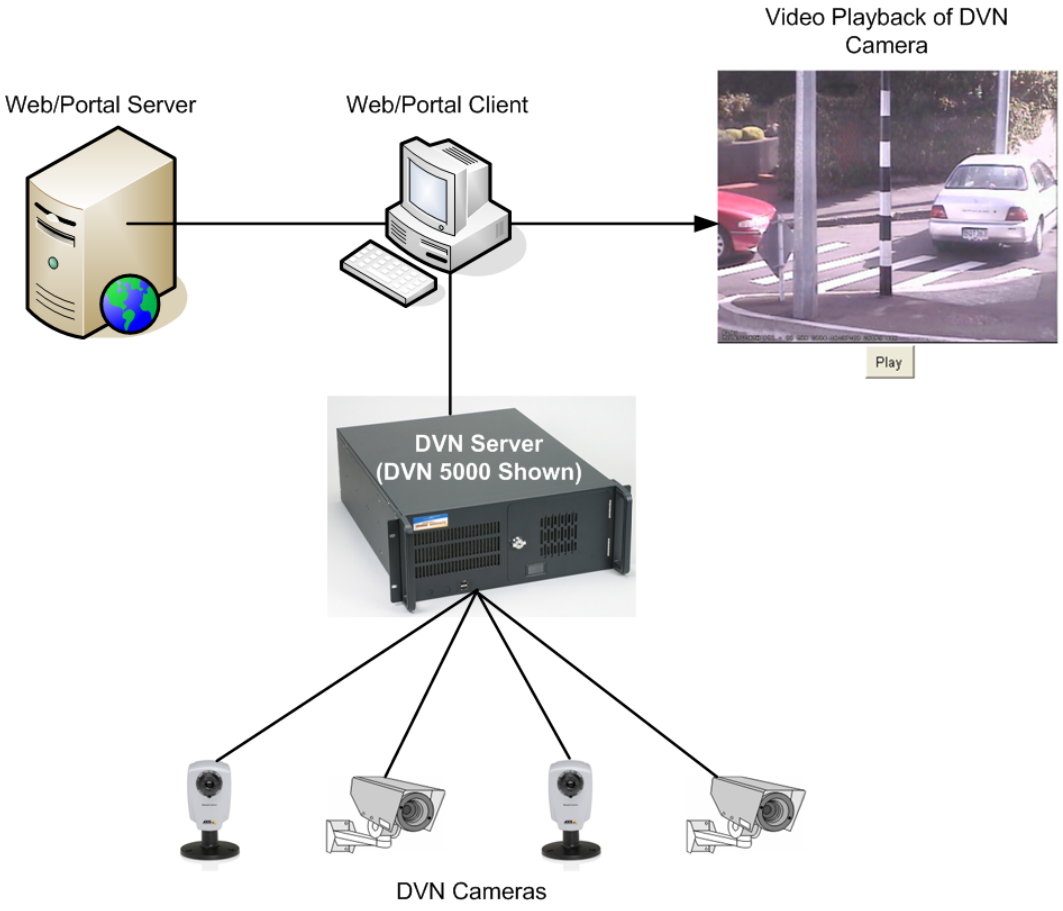


Figure 2-1: Viewing DVN Video Clips from Web/Portal Client

How It Works

The first time a user on a client computer points to the portal page, the user's browser downloads the **SpectivaVisionWEB.cab** file containing **SpectivaWeb.ocx** from the web/portal server and prompts the user to accept installation of the ActiveX control on the client computer. When this occurs, the client creates an instance of SpectivaWeb.ocx in the browser and connects directly to the DVN using that instance of SpectivaWeb.ocx.

The client calls methods in SpectivaWeb.ocx from the web page using Javascript. SpectivaWeb.ocx then communicates with the DVN on the client's behalf.

The portal page works by parsing query string parameters from the URL on the client side, then using the parameter values to control the SpectivaWeb ActiveX control. Presumably, this Uniform Resource Locator (URL) is generated when users click on a link in another page of the portal application. For example, a browser-based surveillance history application might generate a screen containing a list of links to pages detailing suspicious occurrences over the last 7 days. Each of these links would reference a URL similar to the one shown in this example. The information contained in these links might come from an incident database maintained on the web/portal server, or on another server with which the web/portal server communicates. At any rate, the web server itself merely serves the HTML and makes the **SpectivaVisionWEB.cab** file available – it has no role in communicating with the DVN.

Configuration

The **SpectivaVisionWEB.cab** file contains the SpectivaWeb ActiveX control used to display video. Place this file on the web server serving the portal pages, and reference it in the "codebase" parameter of the portal page, as shown in the following sample HTML file. To view the entire sample file, see "Sample HTML File" on page 2-120.

Use the following URL format for the portal page:

http://myPortalServerName/myPortalServerFolderPath/video.html?ipaddress=1.2.3.4&camera=2&start=March 5, 2007 14:52&duration=7

The camera index is zero-based and the duration is in seconds. In addition, the *ipaddress* parameter indicates the address of the DVN server where the video can be located.

NOTES

- *The video clip will play automatically when users first navigate to the page. Once the clip finishes playing, the **Play** button is enabled. Pressing **Play** causes the clip to be played again.*
- *You may use alternate string representations for the start date/time.*
- *Consider how best to handle the username/password for DVN login. These can be hard-coded in the Javascript. They can be passed in the URL as query string parameters; however, this approach has security implications. Instead, consider placing the username and password in a non-persistent cookie before navigating to the HTML page responsible for playing the video clip, which would then extract them from the cookie. More sophisticated integration with enterprise single sign-on infrastructure is also possible, but beyond the scope of this example. See also "Username and Password Prompting" on page 2-123.*
- *The interaction with the SpectivaWeb control occurs in stages and is driven by events coming from the browser and the control. Once the browser indicates that the control has been instantiated, the control's **Connect** method (and some others) is called. Once the control indicates that the connect has occurred, the control's **SetCamera** and **SendImages** methods are called. Finally, once the control indicates that it is ready to handle video playback, the control's **Play** and **PlayFind** methods are called. This is necessary since issuing all the commands at once would result in later commands being ignored if the control was not yet in a state which allowed them to be processed.*
- *You may inline the Javascript.*

Sample HTML File

This section includes sample HTML code of a portal page that could be used to view video from a DVN camera. The web page includes an area for viewing video and a **Play** button to restart the video sequence. Use this code as a guide when developing your own web portal page.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
   <HEAD>
      <title>DVN Sample Video Playback Portal Page</title>
      <script src="play.js" type="text/javascript"></script>
      <script language="JavaScript" for="cameraView"
event="OnCreate()">connectDVNVideo();</script>
```

```
        <script language="JavaScript" for="cameraView" event="OnConnection(_status,
_connection)">initDVNVideo();</script>
        <script language="JavaScript" for="cameraView" event="OnPlayStatus(_status,
_connection))">playDVNVideo();</script>
    </HEAD>
    <body onkeydown="toggleDebugMode()">
        <table>
            <tr>
                <td>
                    <OBJECT id="cameraView"
                        classid="CLSID:BB5DB54E-BAE2-48BF-B0BD-2FD3D9F8EB08"
                        codebase="SpectivaVisionWEB.cab"
                        width="320"
                        height="240" VIEWASTEXT>
                    </OBJECT>
                </td>
            </tr>
            <tr>
                <td align="center">
                    <button id="playButton" type="button" onclick="playDVNVideo()"
disabled="true">Play</button>
                </td>
            </tr>
        </table>
    </body>
</HTML>
```

Sample Javascript File

This section includes sample Javascript code used with the web page and the SpectivaVisionWEB.cab file to view video from a DVN camera. As with the sample HTML code, use this code as a reference when designing your own portal page.

```
var winMain = 0;
var numConnection = 0;
var queryParms = {};

// called after the SpectivaWeb ActiveX control instance has been created in the browser
function connectDVNVideo() {
    getQueryParms();
    try {
        cameraView.createGrid(12,12);
        winMain = cameraView.CreateWin(0,0,12,12);

        // these are hardcoded for now but would be retrieved from LDAP / Sun Access
Manager
        // probably not a good idea to pass these in the query string for security purposes
        var userid = "admin";
        var password = "";

        numConnection = cameraView.Connect(userid, password, queryParms.ipaddress, 0, 5,
95);
```

```
        cameraView.SetMainWindow(winMain);
        cameraView.ChangeResolutionWin(0, winMain);
        cameraView.BringWindowToFront(winMain);
    } catch (e) {
        alert("exception: " + e.message);
    }
}

// called after event indicating successful DVN connection is received
function initDVNVideo() {
    try {
        var result = cameraView.SetCamera(queryParms.camera, numConnection, winMain);
        result = cameraView.SendImages(numConnection, 1);
    } catch (e) {
        alert("exception: " + e.message);
    }
}

// called when video stream is set up and playback is enabled
function playDVNVideo() {
    try {
        playButton.disabled = true;
        cameraView.Play(winMain);
        // note that PlayFind requires start time in number of seconds since 1970, not
milliseonds
        cameraView.PlayFind(winMain, new Date(queryParms.start).getTime() / 1000);
        cameraView.Play(winMain);
        cameraView.style.visibility = "visible";
        window.setTimeout("stopPlay()", parseInt(queryParms.duration) * 1000);
    } catch (e) {
        alert("exception: " + e.message);
    }

}

function stopPlay() {
    cameraView.PlayStill(winMain);
    playButton.disabled = false;
}

// split up the query string and store in an associative array
function getQueryParms() {
    var url = window.location.toString();
    url.match(/\?(.+)$/);
    var params = RegExp.$1;
    var params = params.split("&");
    for(var i=0;i<params.length;i++) {
        var tmp = params[i].split("=");
        queryParms[tmp[0]] = unescape(tmp[1]);
    }
}

// for debugging, in conjunction with Trace checkbox (use CTRL-E to show debugging pane)
function isTraceEnabled() {
```

```
    return ev_evaluator ? ev_evaluator.traceEnabled.checked : false;
}
```

Username and Password Prompting

If the user will be prompted for login credentials when signing onto the portal application, and if subsequent “video-enabled” web pages in the portal application connect to DVNs using the same username and password as used for logging onto the portal, then you may configure the system so that a user will not be reprompted for a username and password when navigating to a video-enabled page.

➤ To configure the system to not reprompt a user to enter his/her username and password:

1. In the HTML for the login page of the web portal, place a declaration for an “invisible” SpectivaWeb ActiveX control, as follows:

   ```
   <object id="VisionWeb1" width="0" height="0"
     classid="CLSID:BB5DB54E-BAE2-48BF-B0BD-2FD3D9F8EB08"
     codebase="SpectivaVisionWEB.cab#version=2,5,7,25">
   </object>
   ```

2. Add some Javascript in the login page for the web portal that calls the *Connect()* method of the SpectivaWeb control, passing the username and password entered by the user, but passing a blank IP address.

   Make sure this Javascript is called after the user enters his username and password, and before the user leaves the page. This will cause the username and password to be stored in a secure memory area for use on subsequent pages in the portal application. Here is an example:

   ```
   VisionWeb1.Connect(form1.usernameField.value, form1.passwordField.value,
   "", 0, 5, 95);
   ```

3. When calling the *Connect()* method on subsequent video-enabled pages of the web portal application, do not pass the username and password, but do pass the IP address of the DVN. This will cause the SpectivaWeb control to retrieve the username and password stored previously, and use them to connect to the DVN. Here is an example:

   ```
   VisionWeb1.Connect("", "", "157.262.15.137", 0, 5, 95);
   ```