

# Overview

---

The SmartCode ViewerX VNC control gives developers full access to the VNC Viewer features using a set of intuitive ActiveX properties and methods. With ViewerX control, developers can easily provide screen sharing and remote control capabilities to their applications.

## Features

- All features that can be found in a standard UltraVNC, TightVNC and RealVNC viewers
- Can work behind HTTP/SOCKS5 proxy servers
- UltraVNC NTLM Windows authentication mode support
- UltraVNC SecureVNC v2.3 and MSCR4 v1.2 DSM encryption plugins support
- UltraVNC Repeater proxy support
- UltraVNC SingleClick server support
- UltraVNC chat support
- TightVNC v1.3 and UltraVNC file transfers support
- Can connect to VNC server in asynchronous mode

## Benefits

- Can be used from any development environment which supports ActiveX.
- Available in both 32 and 64-bit versions.
- Supports Internet Protocol version 6 (IPv6).
- No runtime library dependencies
- Runtime royalty free

## Supported on

- Windows XP, Vista, Windows 7, Windows 8.1, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# Examples

---

## Guide to Examples

- **C#** demonstrates using ViewerX control in a .Net WinForms application.
- **VC** demonstrates using ViewerX control in a MFC based application.
- **HTML** demonstrates using ViewerX control from within an HTML page.

# Frequently Asked Questions

---

## 1. What dependencies are there? What version of the runtime do I need?

ViewerX control file doesn't require any Visual C++ runtime libraries.

## 2. What platforms are supported?

The SmartCode ViewerX control may be used under any Windows platform, namely Windows XP, Vista, Windows 7, Windows Server 2003, Windows Server 2008.

## 2. How to deploy ViewerX to my Web server?

Your web page hosting ViewerX might look like the sample below:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html" />
    <script language="Javascript">
      function OnConnect() {
        window.ViewerX.Connect()
      }
    </script>
  </head>
  <body onload="OnConnect()">
    <object CLASSID="clsid:5220cb21-c88d-11cf-b347-00ae0308947c"
      <PARAM NAME="LPKPath" VALUE="scvncctrl.lpk" />
    </object>

    <object id="ViewerX" height="328" width="662" classid="clsid:5220cb21-c88d-11cf-b347-00ae0308947c"
      CODEBASE="viewerx.cab">
      <param name="HostIP" value="192.168.1.139" />
      <param name="Password" value="123" />
    </object>
  </body>
</html>
```

Note first <object> tag entry. Do not delete it. This is Microsoft's license manager, which provides support for a licensed ViewerX control. It uses scvncctrl.lpk file to read information about ViewerX

license.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.



# Support

---

For further information, visit us at <http://www.s-code.com> or send email to [support@s-code.com](mailto:support@s-code.com). We are always happy to answer your questions.

If you're having trouble using the control, have a bug to report, or wish to suggest an enhancement for a future release send email to [support@s-code.com](mailto:support@s-code.com).

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# Version History

---

[SmartCode ViewerX VNC Viewer version history.](#)

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ViewerX Control Run-Time License

The SmartCode ViewerX VNC control is protected with a design and run-time licensing support. In order to use the control in your application, you must initialize it with the following run-time license key value:

*scviewxlickey*

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# AutoReconnectContinueState Enumeration

---

```
enum AutoReconnectContinueState
{
    //The reconnection process is occurring automatically.
    //This is the default value of the AutoReconnectContinueSt
    ARCS_CONTINUE = 0,
    //The reconnection process has been stopped.
    ARCS_STOP = 1
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ColorDepth Enumeration

---

```
enum ColorDepth
{
    COLOR_FULL = 0,
    COLOR_256  = 1, //256 colors
    COLOR_64   = 2, //64 colors
    COLOR_8    = 3  //8 colors
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ConnectionProxyType Enumeration

---

```
enum ConnectionProxyType
{
    VPT_NONE           = 0, //Direct connection
    VPT_SOCKS5         = 1, //SOCKS5 (no password)
    VPT_HTTP           = 2, //HTTP proxy (no password)
    VPT_ULTRA_REPEATER = 3  //UltraVNC repeater
    VPT_SOCKS5_USRPWD  = 4, //SOCKS5 (with password)
    VPT_HTTP_BASIC     = 5  //HTTP proxy (with password)
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# CursorTrackingMode Enumeration

---

```
enum CursorMode
{
    CM_TRACK_LOCALY    = 0, // Track remote cursor locally
    CM_REMOTE_DEAL     = 1, // Let remote server deal with mouse cur
    CM_DONT_SHOW_REM  = 2  // Don't show remote cursor
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# CursorTrackingMode Enumeration

---

```
enum CursorTrackingMode
{
    VCT_NO_CURSOR      = 0,
    VCT_DOT_CURSOR     = 1,
    VCT_NORMAL_CURSOR  = 2,
    VCT_SMALL_CURSOR   = 3
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.



# EncryptionPluginType Enumeration

---

```
enum EncryptionPluginType
{
    EPT_NONE,          //Plain connection, no encryption
    EPT_MSRC4,         //Use MSRC4 DSM plug-in
    EPT_SECUREVNC     //Use SecureVNC DSM plug-in
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ScreenStretchMode Enumeration

---

```
enum ScreenStretchMode
{
    SSM_NONE    = 0,    //Screen stretch disabled
    SSM_FREE    = 1,    //Resize the remote screen image to fill \
    SSM_ASPECT  = 2     //Scale to as large an image as possible,
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ScreenStretchRenderingQuality Enumeration

---

```
enum ScreenStretchRenderingQuality
{
    // ViewerX uses GDI (StretchBlt) to draw scaled screen bitmap
    SSQ_GDI = 0,

    // ViewerX uses GDI+ to draw scaled screen bitmap
    // SetInterpolationMode(InterpolationModeHighQuality);
    // SetSmoothingMode(SmoothingModeHighQuality);
    SSQ_GDIPLUS_HIGH = 1,

    // ViewerX uses GDI+ to draw scaled screen bitmap
    // SetInterpolationMode(InterpolationModeLowQuality);
    // SetSmoothingMode(SmoothingModeHighSpeed);
    SSQ_GDIPLUS_LOW = 2,
};
```

# VNCEncoding Enumeration

---

```
enum VNCEncoding
{
    RFB_RAW          = 0,
    RFB_RRE          = 2,
    RFB_CORRE        = 4,
    RFB_HEXTILE      = 5,
    RFB_ZLIB         = 6,
    RFB_TIGHT        = 7,
    RFB_ZLIBHEX      = 8,
    RFB_ULTRA        = 9,
    RFB_ZRLE         = 16,
    RFB_ZYWRLE       = 17
};
```

# VncConnectionState Enumeration

---

```
enum VncConnectionState
{
    VCS_DISCONNECTED = 0,
    VCS_CONNECTED    = 1,
    VCS_CONNECTING   = 2,
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer2::AltKeyPressed Property

---

Sets or retrieves the value indicating whether Alt key is pressed at the VNC server side.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::get_AltKeyPressed(VARIANT_BOOL* pbPr  
HRESULT ISmartCodeVNCViewer2::put_AltKeyPressed(VARIANT_BOOL bPres
```

## Parameters

*pbAllow*

Pointer to a variable of type **VARIANT\_BOOL** that receives Alt key status.

*bAllow*

**VARIANT\_BOOL** that presses or depresses Alt key at the VNC server side.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::AdvancedSettings Property

---

Retrieves an instance of the IScVxAdvancedSettings interface.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_AdvancedSettings(IScVxAdvancedSet
```

## Parameters

*ppAdvSettings*

Address of the current control's IScVxAdvancedSettings interface.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::CacheEncoding Property

---

Sets or retrieves the value indicating whether Cache encoding is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_CacheEncoding(VARIANT_BOOL* pbAllow)  
HRESULT ISmartCodeVNCViewer3::put_CacheEncoding(VARIANT_BOOL bAllow)
```

## Parameters

*pbAllow*

Pointer to a variable of type **VARIANT\_BOOL** that receives Cache encoding status.

*bAllow*

**VARIANT\_BOOL** that enables or disables Cache encoding.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::Connected Property

---

Returns a value indicating whether a VNC control is connected.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_Connected(VARIANT_BOOL* pbConnect
```

## Parameters

*pbConnected*

Pointer to a variable of type **VARIANT\_BOOL** that presents whether a VNC control is connected.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::ColorDepth Property

---

Sets or gets the color depth of the screen image.

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_ColorDepth(ColorDepth* pnColorDe  
HRESULT ISmartCodeVNCViewer3::put_ColorDepth(ColorDepth nColorDept
```

## Parameters

*pnColorDepth*

Pointer to a variable of type **ColorDepth** that receives a color depth of the screen image.

*nColorDepth*

**ColorDepth** that specifies a color depth that should be used to draw the screen image.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# ISmartCodeVNCViewer::CopyRect Property

---

Sets or retrieves the value indicating whether CopyRect encoding is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_CopyRect(VARIANT_BOOL* pbAllow);  
HRESULT ISmartCodeVNCViewer::put_CopyRect(VARIANT_BOOL bAllow);
```

## Parameters

*pbAllow*

Pointer to a variable of type **VARIANT\_BOOL** that receives CopyRect encoding status.

*bAllow*

**VARIANT\_BOOL** that enables or disables CopyRect encoding.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ConnectingText Property

---

This method sets or gets the value of the ConnectingText property, the text that appears centered in the control while the control is connecting.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ConnectingText(BSTR* pstrText);  
HRESULT ISmartCodeVNCViewer::put_ConnectingText(BSTR strText);
```

## Parameters

*pstrText*

Pointer to the value of the ConnectingText property.

*strText*

**BSTR** that specifies the string value the ConnectingText property is to be set to.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

An example of connection text is "Connecting to server...". You can use %s combination in your text. It will be replaced with the current server address.

# ISmartCodeVNCViewer::CustomCompressionProperty

---

Sets or retrieves the value indicating whether Custom Compression is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_CustomCompression(VARIANT_BOOL* p  
HRESULT ISmartCodeVNCViewer::put_CustomCompression(VARIANT_BOOL bA
```

## Parameters

*pbAllow*

Pointer to a variable of type **VARIANT\_BOOL** that receives Custom Compression status.

*bAllow*

**VARIANT\_BOOL** that enables or disables Custom Compression encoding.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::CustomCompressionProperty

---

Sets or retrieves custom compression level used during connection to VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_CustomCompressionLevel(long* pnLevel)  
HRESULT ISmartCodeVNCViewer::put_CustomCompressionLevel(long nLevel)
```

## Parameters

*pnLevel*

Pointer to a variable of type **long** that receives custom compression level.

*nLevel*

**long** that specifies custom compression level used during connection to VNC server. The value must be in range from 1 to 9.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer2::CtrlKeyPressed Property

---

Sets or retrieves the value indicating whether Ctrl key is pressed at the VNC server side.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::get_CtrlKeyPressed(VARIANT_BOOL* pbF  
HRESULT ISmartCodeVNCViewer2::put_CtrlKeyPressed(VARIANT_BOOL bPre
```

## Parameters

*pbAllow*

Pointer to a variable of type **VARIANT\_BOOL** that receives Ctrl key status.

*bAllow*

**VARIANT\_BOOL** that presses or depresses Ctrl key at the VNC server side.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::DisableClipboard Property

---

Sets or returns a value indicating whether clipboard .

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_DisableClipboard(VARIANT_BOOL* pbClip)
HRESULT ISmartCodeVNCViewer::put_DisableClipboard(VARIANT_BOOL bClip)
```

## Parameters

*pbClip*

Pointer to a variable of type **VARIANT\_BOOL** that receives clipboard transfer status.

*bClip*

**VARIANT\_BOOL** that enables or disables clipboard transfer.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::DisconnectedText Property

---

This method sets or gets the value of the DisconnectedText property, the text that appears centered in the control while the control is in disconnected.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_DisconnectedText(BSTR* pstrText);  
HRESULT ISmartCodeVNCViewer::put_DisconnectedText(BSTR strText);
```

## Parameters

*pstrText*

Pointer to the value of the DisconnectedText property.

*strText*

**BSTR** that specifies the string value the DisconnectedText property is to be set to.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

An example of connection text is "SmartCode VNC Viewer - Disconnected".

# ISmartCodeVNCViewer::EmulateThreeButton Property

---

Sets or retrieves the value indicating whether 3 mouse button emulation is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_EmulateThreeButton(VARIANT_BOOL*  
HRESULT ISmartCodeVNCViewer::put_EmulateThreeButton(VARIANT_BOOL b
```

## Parameters

*pbEmulate*

Pointer to a variable of type **VARIANT\_BOOL** that receives 3 mouse button emulation status.

*bEmulate*

**VARIANT\_BOOL** that enables or disables 3 mouse button emulation.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::EnableAutoRecon Property

---

Specifies whether to enable the client control to reconnect automatically to a session in the event of a network disconnection.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::get_EnableAutoReconnect(VARIANT_BOOL  
HRESULT ISmartCodeVNCViewer2::put_EnableAutoReconnect(VARIANT_BOOL
```

## Parameters

### *pbEnable*

Pointer to a variable of type `VARIANT_BOOL` that receives the automatic reconnection status: `VARIANT_TRUE` to enable and `VARIANT_FALSE` otherwise.

### *bEnable*

Set to `VARIANT_TRUE` to enable automatic reconnection, and to `VARIANT_FALSE` to disable it. The default is `VARIANT_TRUE`.

## Return Value

Returns `S_OK` if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Encoding Property

---

Sets or returns an encoding used to connect to server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_Encoding(VNCencoding* pnEncoding)  
HRESULT ISmartCodeVNCViewer::put_Encoding(VNCencoding nEncoding);
```

## Parameters

*pnEncoding*

Pointer to a variable of type **VNCencoding** that receives an encoding used to connect to server.

*nEncoding*

**VNCencoding** that specifies connection encoding.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::EncryptionPlugin Property

---

This property allows you to specify the encryption algorithm (your choices include no encryption, MSRC4, or SecureVNC DSM plug-ins).

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_EncryptionPlugin(E  
HRESULT ISmartCodeVNCViewer3::put_EncryptionPlugin(E
```

## See Also

[EncryptionPluginType](#)

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::FullScreen Property

---

Sets or retrieves a value indicating whether the control is in full-screen mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_FullScreen(VARIANT_BOOL* pbFull);  
HRESULT ISmartCodeVNCViewer::put_FullScreen(VARIANT_BOOL bFull);
```

## Parameters

*pbFull*

Pointer to a variable of type **VARIANT\_BOOL** that receives full screen attribute.

*bFull*

**VARIANT\_BOOL** that enables or disables a full screen mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::HostIP Property

---

Sets or retrieves the remote VNC server address.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_HostIP(BSTR* pstrHostIP);  
HRESULT ISmartCodeVNCViewer::put_HostIP(BSTR strHostIP);
```

## Parameters

*pstrHostIP*

Pointer to a variable of type **BSTR** that receives the address of target VNC server.

*strHostIP*

**BSTR** that specifies the address of target VNC server.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::JPEGCompression Property

---

Sets or retrieves the value indicating whether JPEG compression is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_JPEGCompression(VARIANT_BOOL* pbAllow)  
HRESULT ISmartCodeVNCViewer::put_JPEGCompression(VARIANT_BOOL bAllow)
```

## Parameters

*pbAllow*

Pointer to a variable of type **VARIANT\_BOOL** that receives JPEG compression status.

*bAllow*

**VARIANT\_BOOL** that enables or disables JPEG compression encoding.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::JPEGCompressionLevel Property

---

Sets or retrieves custom compression level used during connection to VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_JPEGCompressionLevel(long* pnLevel)  
HRESULT ISmartCodeVNCViewer::put_JPEGCompressionLevel(long nLevel)
```

## Parameters

*pnLevel*

Pointer to a variable of type **long** that receives JPEG compression level.

*nLevel*

**long** that specifies JPEG compression level used during connection to VNC server. The value must be in range from 0 to 9.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Listening Property

---

Returns a value indicating whether a VNC control is listening for incoming VNC server connections.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_Listening(VARIANT_BOOL* pbListeni
```

## Parameters

*pbConnected*

Pointer to a variable of type **VARIANT\_BOOL** that prepresents whether a VNC control is listening for connections.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ListeningText Property

---

This method sets or gets the value of the ListeningText property, the text that appears centered in the control while the control is listening for server connections.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ListeningText(BSTR* pstrText);  
HRESULT ISmartCodeVNCViewer::put_ListeningText(BSTR strText);
```

## Parameters

*pstrText*

Pointer to the value of the ListeningText property.

*strText*

**BSTR** that specifies the string value the ListeningText property is to be set to.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

An example of connection text is "Listening for incoming connections...".

You can use %i combination in your text. It will be replaced with the current listening port value.

# ISmartCodeVNCViewer::ListenPort Property

---

Gets or sets the port to use when ViewerX is listening for incoming VNC server connections.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ListenPort(long* pnPort);  
HRESULT ISmartCodeVNCViewer::put_ListenPort(long nPort);
```

## Parameters

*pnPort*

Pointer to a variable of type **long** that receives the port ViewerX is listens for incoming connections.

*nPort*

**long** that specifies the port to listen for incoming connections.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::LocalCursor Property

---

Sets or retrieves mouse cursor handling mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_LocalCursor(CursorTrackingMode* p  
HRESULT ISmartCodeVNCViewer::put_LocalCursor(CursorTrackingMode n1
```

## Parameters

*pnTrack*

Pointer to a variable of type **CursorTrackingMode** that receives mouse cursor handling mode.

*nTrack*

**CursorTrackingMode** that specifies mouse cursor handling mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::MessageBoxes Property

---

Sets or retrieves a value indicating whether the control would show any message boxes.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_MessageBoxes(VARIANT_BOOL* pbMsg)  
HRESULT ISmartCodeVNCViewer::put_MessageBoxes(VARIANT_BOOL bMsg);
```

## Parameters

*pbMsg*

Pointer to a variable of type **VARIANT\_BOOL** that receives message boxes mode attribute.

*bMsg*

**VARIANT\_BOOL** that enables or disables message boxes mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::MouseCursorMode Property

---

Sets or retrieves mouse cursor handling mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_MouseCursorMode(CursorMode* pnMode)  
HRESULT ISmartCodeVNCViewer::put_MouseCursorMode(CursorMode nMode)
```

## Parameters

*pnMode*

Pointer to a variable of type **CursorMode** that receives mouse cursor handling mode.

*nMode*

**CursorMode** that specifies mouse cursor handling mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::MsDomain Property

---

Sets or retrieves domain name used for Windows NTLM authentication.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_MsDomain(BSTR* pstrDomain);  
HRESULT ISmartCodeVNCViewer::put_MsDomain(BSTR strDomain);
```

## Parameters

*pstrDomain*

Pointer to a variable of type **BSTR** that receives domain name.

*strDomain*

**BSTR** that specifies domain name.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::MsPassword Property

---

Sets or retrieves user password used for Windows NTLM authentication.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_MsPassword(BSTR* pstrPassword);  
HRESULT ISmartCodeVNCViewer::put_MsPassword(BSTR strPassword);
```

## Parameters

*pstrPassword*

Pointer to a variable of type **BSTR** that receives Windows user password.

*strPassword*

**BSTR** that specifies Windows user password.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::MsUser Property

---

Sets or retrieves user name used for Windows NTLM authentication.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_MsUser(BSTR* pstrUser);  
HRESULT ISmartCodeVNCViewer::put_MsUser(BSTR strUser);
```

## Parameters

*pstrUser*

Pointer to a variable of type **BSTR** that receives Windows user name.

*strUser*

**BSTR** that specifies Windows user name.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::OuterBackgroundColor Property

---

Sets the outer background color. This background color is used to paint the background when the ViewerX size is larger than the remote screen. By default the COLOR\_APPWORKSPACE system color is used to paint the outer background.

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_OuterBackgroundColor(COLORREF*  
HRESULT ISmartCodeVNCViewer3::put_OuterBackgroundColor(COLORREF c
```

## Parameters

*pcrColor*

Pointer to a variable of type **long** that receives RGB color value used to paint the outer background area.

*crColor*

**long** that specifies the RGB color to use paint the outer background area.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Password Property

---

Gets or sets the password to use when authenticating the client against VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_Password(BSTR* pstrPassword);  
HRESULT ISmartCodeVNCViewer::put_Password(BSTR strPassword);
```

## Parameters

*pstrPassword*

Pointer to a variable of type **BSTR** that receives the password.

*strPassword*

**BSTR** that specifies the password to use when authenticating the client.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Port Property

---

Gets or sets the port to use when connecting to VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_Port(long* pnPort);  
HRESULT ISmartCodeVNCViewer::put_Port(long nPort);
```

## Parameters

*pnPort*

Pointer to a variable of type **long** that receives the port of target VNC server.

*nPort*

**long** that specifies the port of VNC server.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ProxyIP Property

---

Sets or retrieves the proxy server address to use when connecting to a VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ProxyIP(BSTR* pstrProxyIP);  
HRESULT ISmartCodeVNCViewer::put_ProxyIP(BSTR strProxyIP);
```

## Parameters

*pstrProxyIP*

Pointer to a variable of type **BSTR** that receives the address of a proxy server.

*strProxyIP*

**BSTR** that specifies the address of a proxy server.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ProxyPassword Property

---

Gets or sets the password to use when authenticating the client against proxy server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ProxyPassword(BSTR* pstrPassword)  
HRESULT ISmartCodeVNCViewer::put_ProxyPassword(BSTR strPassword);
```

## Parameters

*pstrPassword*

Pointer to a variable of type **BSTR** that receives the password.

*strPassword*

**BSTR** that specifies the password to use when authenticating the client.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ProxyType Property

---

Sets or returns an type of proxy server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ProxyType(ConnectionProxyType* p1  
HRESULT ISmartCodeVNCViewer::put_ProxyType(ConnectionProxyType type
```

## Parameters

*p1*

Pointer to a variable of type [ConnectionProxyType](#) that receives the type of proxy server.

*type*

[ConnectionProxyType](#) that specifies proxy server type.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::ProxyUser Property

---

Sets or retrieves user name used to authenticate against proxy server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ProxyUser(BSTR* pstrUser);  
HRESULT ISmartCodeVNCViewer::put_ProxyUser(BSTR strUser);
```

## Parameters

*pstrUser*

Pointer to a variable of type **BSTR** that receives user name.

*strUser*

**BSTR** that specifies proxy server user name.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::RequestSharedSes Property

---

Sets or retrieves the value indicating whether "request shared session" mode is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_RequestSharedSession(VARIANT_BOOL  
HRESULT ISmartCodeVNCViewer::put_RequestSharedSession(VARIANT_BOOL
```

## Parameters

*pbShared*

Pointer to a variable of type **VARIANT\_BOOL** that receives "request shared session" mode status.

*bShared*

**VARIANT\_BOOL** that enables or disables "request shared session" mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer2::RemoteInputEnabled Property

---

Enables or disables remote input. UltraVNC server specific feature.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::get_RemoteInputEnabled(VARIANT_BOOL *  
HRESULT ISmartCodeVNCViewer2::put_RemoteInputEnabled(VARIANT_BOOL
```

## Parameters

*pbEnable*

Pointer to a variable of type **VARIANT\_BOOL** that receives remote input status.

*bEnable*

**VARIANT\_BOOL** that enables or disables remote input.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::RestrictPixel Property

---

Deprected: Use the [ColorDepth](#) property instead!

Sets or retrieves the value indicating whether "restrict pixel", in other words 8-bit color mode is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_RestrictPixel(VARIANT_BOOL* pbRes  
HRESULT ISmartCodeVNCViewer::put_RestrictPixel(VARIANT_BOOL bRestr
```

## Parameters

*pbRestrict*

Pointer to a variable of type **VARIANT\_BOOL** that receives 8-bit color mode status.

*bRestrict*

**VARIANT\_BOOL** that enables or disables 8-bit color mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ScaleDen Property

---

Sets or retrieves scale denominator value.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ScaleDen(long* pnScale);  
HRESULT ISmartCodeVNCViewer::put_ScaleDen(long nScale);
```

## Parameters

*pnScale*

Pointer to a variable of type **long** that receives the scale denominator value.

*nScale*

**long** that specifies the scale denominator value.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ScaleEnable Property

---

Sets or retrieves status of the standard VNC viewer screen scaling mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ScaleEnable(VARIANT_BOOL* pbScale);  
HRESULT ISmartCodeVNCViewer::put_ScaleEnable(VARIANT_BOOL bScale);
```

## Parameters

*pbScale*

Pointer to a variable of type **VARIANT\_BOOL** that screen scaling status.

*bScale*

**VARIANT\_BOOL** that enables or disables screen scaling.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ScaleNum Property

---

Sets or retrieves scale numerator value.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ScaleNum(long* pnScale);  
HRESULT ISmartCodeVNCViewer::put_ScaleNum(long nScale);
```

## Parameters

*pnScale*

Pointer to a variable of type **long** that receives the scale numerator value.

*nScale*

**long** that specifies the scale numerator value.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ScreenBitmap Property

---

Returns HBITMAP handle of the screen bitmap.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ScreenBitmap(long* pnScreenBitmap)
```

## Parameters

*pnScreenBitmapHandle*

Pointer to a variable of type **long** that receives HBITMAP handle of the screen bitmap.

## Remarks

You **must not delete** the bitmap handle returned by the ScreenBitmap property. This handle is used by ViewerX and mustn't be modified by the hosting application.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::ScreenHeight Property

---

Retrieves height of remote screen.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ScreenHeight(long* pnScreenHeight)
```

## Parameters

*pnScreenHeight*

Pointer to a variable of type **long** that receives height of remote screen.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ScreenWidth Property

---

Retrieves width of remote screen.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ScreenWidth(long* pnScreenWidth);
```

## Parameters

*pnScreenWidth*

Pointer to a variable of type **long** that receives width of remote screen.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::StretchMode Property

---

Sets or returns a client side screen stretching mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_StretchMode(ScreenStretchMode* pn  
HRESULT ISmartCodeVNCViewer::put_StretchMode(ScreenStretchMode mode
```

## Parameters

*pmode*

Pointer to a variable of type **ScreenStretchMode** that receives screen stretching mode.

*mode*

**ScreenStretchMode** that specifies screen stretching mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# ISmartCodeVNCViewer3::ScreenStretchRen Property

---

Sets or returns a client side screen stretching mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_ScreenStretchRenderingQuality(Sc  
HRESULT ISmartCodeVNCViewer3::put_ScreenStretchRenderingQuality(Sc
```

## Parameters

*quality*

**ScreenStretchRenderingQuality** that specifies screen stretching rendering engine used to scale the remote screen bitmap.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::SwapMouseButtons Property

Sets or retrieves the value indicating whether mouse buttons swapping is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_SwapMouseButtons(VARIANT_BOOL* pb  
HRESULT ISmartCodeVNCViewer::put_SwapMouseButtons(VARIANT_BOOL bSw
```

## Parameters

*pbSwap*

Pointer to a variable of type **VARIANT\_BOOL** that receives mouse button swapping status.

*bSwap*

**VARIANT\_BOOL** that enables or disables mouse button swapping.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::TopLevelParent Property

---

Sets or retrieves HWND handle use as a parent window for ViewerX message boxes and dialogs.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_TopLevelParent(long* phParent);  
HRESULT ISmartCodeVNCViewer::put_TopLevelParent(long hParent);
```

## Parameters

*phParent*

Pointer to a variable of type **long** that receives the address of target VNC server.

*hParent*

**long** that specifies the address of target VNC server.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::UltraVNCSecurity\_Property

---

Retrieves an instance of the IScVxUltraSecurity\_MSRC4 interface. The interface allows developers to control UltraVNC MSRC4 DSM encryption plug-in related settings.

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_UltraVNCSecurity_MSRC4(IScVxUltr
```

## Parameters

*ppSecurity*

Address of the current control's IScVxUltraSecurity\_MSRC4 interface.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer3::UltraVNCSecurity\_Property

---

Retrieves an instance of the IScVxUltraSecurity\_SecureVNC interface. The interface allows developers to control UltraVNC SecureVNC DSM encryption plug-in related settings.

## Syntax

```
HRESULT ISmartCodeVNCViewer3::get_UltraVNCSecurity_SecureVNC(IScVx
```

## Parameters

*ppSecurity*

Address of the current control's IScVxUltraSecurity\_SecureVNC interface.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# ISmartCodeVNCViewer::ViewOnly Property

---

Sets or retrieves the value indicating whether "view mode" mode is enabled.

## Syntax

```
HRESULT ISmartCodeVNCViewer::get_ViewOnly(VARIANT_BOOL* pbViewOnly)  
HRESULT ISmartCodeVNCViewer::put_ViewOnly(VARIANT_BOOL bViewOnly);
```

## Parameters

*pbViewOnly*

Pointer to a variable of type **VARIANT\_BOOL** that receives "view mode" mode status.

*bViewOnly*

**VARIANT\_BOOL** that enables or disables "view mode" mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Connect Function

---

Initiates a connection using the properties currently set on the control.

## Syntax

```
HRESULT ISmartCodeVNCViewer::Connect();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

The only required property is the server address.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::ConnectEx Function

---

Initiates a connection to the VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::ConnectEx(BSTR strIP, long nPort, BSTR
```

## Parameters

*strIP*

VNC server IP address or hostname.

*nPort*

Connection port.

*strPassword*

Password used for VNC server authentication.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ConnectAsync Function

---

Initiates an asynchronous connection using the properties currently set on the control.

## Syntax

```
HRESULT ISmartCodeVNCViewer::ConnectAsync();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

The only required property is the server address.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::ConnectAsyncEx Function

---

Initiates an asynchronous connection to the VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::ConnectAsyncEx(BSTR strIP, long nPort
```

## Parameters

*strIP*

VNC server IP address or hostname.

*nPort*

Connection port.

*strPassword*

Password used for VNC server authentication.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Disconnect Function

---

Disconnects the active connection.

## Syntax

```
HRESULT ISmartCodeVNCViewer::Disconnect();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::GetConnectionState Function

---

Returns state of active connection.

## Syntax

```
HRESULT ISmartCodeVNCViewer::GetConnectionState(VncConnectionState
```

## Parameters

*pConState*

Pointer to a variable of type [VncConnectionState](#) that receives active connection state.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# ISmartCodeVNCViewer::GetScreenBitmapScaled Function

---

Resizes the screen image to the specified width and height and returns HBITMAP handle of the scaled bitmap.

## Syntax

```
HRESULT ISmartCodeVNCViewer::GetScreenBitmapScaled(1
```

## Parameters

*nWigth*

Width of the scaled bitmap in pixels.

*nHeight*

Height of the scaled bitmap in pixels.

*phBitmapHandle*

Pointer to a variable of type **long** that receives HBITMAP handle of the scaled bitmap.

## Remarks

You are responsible for deleting the HBITMAP handle returned by the GetScreenBitmapScaled method. You must release this handle using the GDI [DeleteObject](#) method.



# ISmartCodeVNCViewer::Listen Function

---

Tells ViewerX to start listen for incoming VNC server connections.

## Syntax

```
HRESULT ISmartCodeVNCViewer::Listen();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

The only required property is a listening VNC port.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::ListenEx Function

---

Tells ViewerX to start listen for incoming VNC server connections.

## Syntax

```
HRESULT ISmartCodeVNCViewer::ListenEx(long nListeningPort);
```

## Parameters

*nPort*

Port to listen for incoming connections.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer2::OpenChat Function

---

Opens VNC chat window. At the moment only UltraVNC chat is supported.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::OpenChat();
```

## Remarks

You can use [IScVxCapabilities::Chat](#) property to check if the remote VNC server supports VNC chat feature.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer2::OpenFileTransfer Function

---

Opens VNC file transfer window. This feature works with TightVNC v1.3 and UltraVNC based servers only.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::OpenFileTransfer();
```

## Remarks

You can use [IScVxCapabilities::FileTransfer](#) property to check if the remote VNC server supports VNC file transfer feature.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::RequestRefresh Function

---

Requests full screen refresh.

## Syntax

```
HRESULT ISmartCodeVNCViewer::RequestRefresh();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::SendCAD Function

---

Send Ctrl+Alt+Del keystroke to the VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::SendCAD();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer2::SelectSingleWindow Function

---

Selects single window for broadcasting by a remote UltraVNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::SelectSingleWindow(VARIANT_BOOL bSel
```

## Parameters

*bSelect*

If *bSelect* equals TRUE, ViewerX changes to 'select single window mode' and switches back to normal mode after user select a window. If *bSelect* equals FALSE, then the whole remote desktop will be shown.

## Remarks

You can use [IScVxCapabilities::SelectSingleWindow](#) property to check if the remote VNC server supports switching between monitors.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# ISmartCodeVNCViewer2::SendCtrlEsq Function

---

Send Ctrl+Esq keystroke to the VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::SendCtrlEsq();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.



# ISmartCodeVNCViewer2::SendCustomKey Function

---

Sends a keycode to a remote server.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::SendCustomKey(long IKeyCode);
```

## Parameters

*IKeyCode*

Key code of a key combination, which will be sent to server.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# ISmartCodeVNCViewer2::SendCustomKeyEvent Function

---

Sends key up or key down notifications to a remote VNC server.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::SendCustomKeyEvent(long IKeyCode, VARIANT bKeyDownEvent)
```

## Parameters

*IKeyCode*

Specifies the scan code of the key, which will be sent to server.

*bKeyDownEvent*

VARIANT\_TRUE - key down event, VARIANT\_FALSE - key up event.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

The table below lists the scan codes supported by Windows VNC servers.

NOTE: This list may be incomplete. Please refer to the following document for more information about Windows keyboard scan codes:

<http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/scancode.doc>

XK_VoidSymbol	0xFFFFFFFF
XK_space	0x020
XK_BackSpace	0xFF08
XK_Tab	0xFF09
XK_Linefeed	0xFF0A

XK_Clear	0xFF0B
XK_Return	0xFF0D
XK_Pause	0xFF13
XK_Scroll_Lock	0xFF14
XK_Sys_Req	0xFF15
XK_Escape	0xFF1B
XK_Delete	0xFFFF
XK_Print	0xFF61
XK_dead_grave	0xFE50
XK_dead_acute	0xFE51
XK_dead_circumflex	0xFE52
XK_dead_tilde	0xFE53
XK_dead_diaeresis	0xFE57
XK_Home	0xFF50
XK_Left	0xFF51
XK_Up	0xFF52
XK_Right	0xFF53
XK_Down	0xFF54
XK_Page_Up	0xFF55
XK_Page_Down	0xFF56
XK_End	0xFF57
XK_Begin	0xFF58
XK_Select	0xFF60
XK_Print	0xFF61
XK_Execute	0xFF62
XK_Insert	0xFF63
XK_Cancel	0xFF69
XK_Help	0xFF6A
XK_Break	0xFF6B
XK_Num_Lock	0xFF7F
XK_KP_Space	0xFF80
XK_KP_Tab	0xFF89

XK_KP_Enter	0xFF8D
XK_KP_Home	0xFF95
XK_KP_Left	0xFF96
XK_KP_Up	0xFF97
XK_KP_Right	0xFF98
XK_KP_Down	0xFF99
XK_KP_Prior	0xFF9A
XK_KP_Page_Up	0xFF9A
XK_KP_Next	0xFF9B
XK_KP_Page_Down	0xFF9B
XK_KP_End	0xFF9C
XK_KP_Begin	0xFF9D
XK_KP_Insert	0xFF9E
XK_KP_Delete	0xFF9F
XK_KP_Equal	0xFFBD
XK_KP_Multiply	0xFFAA
XK_KP_Add	0xFFAB
XK_KP_Separator	0xFFAC
XK_KP_Subtract	0xFFAD
XK_KP_Decimal	0xFFAE
XK_KP_Divide	0xFFAF
XK_KP_0	0xFFB0
XK_KP_1	0xFFB1
XK_KP_2	0xFFB2
XK_KP_3	0xFFB3
XK_KP_4	0xFFB4
XK_KP_5	0xFFB5
XK_KP_6	0xFFB6
XK_KP_7	0xFFB7
XK_KP_8	0xFFB8
XK_KP_9	0xFFB9
XK_F1	0xFFBE

XK_F2	0xFFBF
XK_F3	0xFFC0
XK_F4	0xFFC1
XK_F5	0xFFC2
XK_F6	0xFFC3
XK_F7	0xFFC4
XK_F8	0xFFC5
XK_F9	0xFFC6
XK_F10	0xFFC7
XK_F11	0xFFC8
XK_F12	0xFFC9
XK_F13	0xFFCA
XK_F14	0xFFCB
XK_F15	0xFFCC
XK_F16	0xFFCD
XK_F17	0xFFCE
XK_F18	0xFFCF
XK_F19	0xFFD0
XK_F20	0xFFD1
XK_F21	0xFFD2
XK_F22	0xFFD3
XK_F23	0xFFD4
XK_F24	0xFFD5
XK_Shift_L	0xFFE1
XK_Shift_R	0xFFE2
XK_Control_L	0xFFE3
XK_Control_R	0xFFE4
XK_Caps_Lock	0xFFE5
XK_Shift_Lock	0xFFE6
XK_Meta_L	0xFFE7
XK_Meta_R	0xFFE8
XK_Alt_L	0xFFE9

XK\_Alt\_R

0xFFEA

## Example

The example below shows how to send Ctrl+Alt+Del combination to a remote VNC server:

```
viewerX.SendCustomKeyEx(65507, true);  
viewerX.SendCustomKeyEx(65513, true);  
viewerX.SendCustomKeyEx(65535, true);  
viewerX.SendCustomKeyEx(65535, false);  
viewerX.SendCustomKeyEx(65513, false);  
viewerX.SendCustomKeyEx(65507, false);
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::SetDormant Function

---

Sets a dormant mode. When the control is in a dormant mode, no screen updates will be send from server.

## Syntax

```
HRESULT ISmartCodeVNCViewer::SetDormant(VARIANT_BOOL bDormant);
```

## Parameters

- *bDormant*  
Enables or disables dormant mode.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ShowConnectionInfo Function

---

Shows a dialog box with an information about active connection.

## Syntax

```
HRESULT ISmartCodeVNCViewer::ShowConnectionInfo();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.



# ISmartCodeVNCViewer::StopListen Function

---

Cancels listening VNC viewer mode.

## Syntax

```
HRESULT ISmartCodeVNCViewer::StopListen();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer2::SwitchMultiMonitor Function

---

Allows to switch between monitors if the remote UltraVNC server runs on a computers with multiple monitors attached.

## Syntax

```
HRESULT ISmartCodeVNCViewer2::SwitchMultiMonitor();
```

## Remarks

You can use [IScVxCapabilities::SwitchMultiMonitor](#) property to check if the remote VNC server supports switching between monitors.

## Return Value

Return Value returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::AuthenticationFailed Event

---

Fired in case if ViewerX has failed to authenticate with VNC server. The caller can use this even to suppress the default password prompt dialog box as well as the "authentication failed" message box.

## Syntax

```
HRESULT AuthenticationFailed(VARIANT_BOOL* pbCancelAndDontPromptFor
```

## Parameters

*pbCancelAndDontPromptForPassword*

Pointer to a variable of type **VARIANT\_BOOL\*** that allows the cancellation of the default password prompt dialog. If **VARIANT\_TRUE**, the password dialog and the "authentication failed" message boxes are suppressed; **VARIANT\_FALSE** allows those dialog boxes to be displayed.  
Default value: **VARIANT\_FALSE**

## Remarks

If attempt to establish connection was performed using asynchronous connection with either [ConnectAsync\(\)](#) or [ConnectAsyncEx\(\)](#) methods, then the event sink callback function will be executed in the non-GUI thread.

Do not attempt to re-establish a connection inside the AuthenticationFailed event sink callback function body.

## Return Value

Returns **S\_OK** if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Connected Event

---

Fired when connection with server has been established.

## Syntax

```
HRESULT Connected();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::Connecting Event

---

Called when the client control begins connecting to a server in response to a call to [ISmartCodeVNCViewer::ConnectAsync](#) or [ISmartCodeVNCViewer::ConnectAsyncEx](#).

## Syntax

```
HRESULT Connecting();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::ConnectionAcceptedEvent

---

Fired when listening viewer has accepted connection from a VNC server.

## Syntax

```
HRESULT ConnectionAccepted(BSTR strServerAddress);
```

## Parameters

*strServerAddress*

**BSTR** that specifies address of remote VNC server connected to the VNC control.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::Disconnected Event

---

Called when the client control has been disconnected from the VNC server.

## Syntax

```
HRESULT Disconnected();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::OnAutoReconnectir Event

---

Fired when a client is in the process of automatically reconnecting to a VNC server.

## Syntax

```
HRESULT OnAutoReconnecting(long attemptCount, enum AutoReconnectCo
```

## Parameters

### *attemptCount*

Number of attempts that have been made in the current automatic reconnection process. This count increases by one for each attempt made.

### *pArcContinueStatus*

Pointer to a returned code of type [AutoReconnectContinueState](#) specifying the state of the automatic reconnection process. This code can be reset to change the state of the current automatic reconnection process.

## Value

Returns S\_OK if successful, or an error value otherwise.

## Remarks

Implement this method in your event sink to receive notification that the control is reestablishing a connection with a VNC server.

When the state of the automatic reconnection process is changed by setting the value of the *pArcContinueStatus* parameter to ARCS\_CONTINUE, this method functions in a purely advisory mode. Containers can listen to this event for notifications that the automatic reconnection process is proceeding. The control will automatically



keep trying to re-establish a connection based on its own internal timing and attempt counts. This method is called during each automatic reconnection attempt in order to notify the container.

When the state of the automatic reconnection process is changed by setting the value of the `pArcContinueStatus` parameter to `ARCS_STOP`, the current automatic reconnection attempt will be terminated, a disconnect notification will be sent to the container, and no further automatic reconnect notifications will be issued.

**Note** Use the [EnableAutoReconnect](#) property to enable or disable automatic reconnection.

# ISmartCodeVNCViewer::OnChatMessageSe Event

---

Fired when ViewerX sends a chat message.

## Syntax

```
HRESULT OnChatMessageSend(BSTR messageText);
```

## Parameters

*messageText*  
The message text.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::OnChatMessageReceived

---

Fired when ViewerX receives a chat message.

## Syntax

```
HRESULT OnChatMessageReceived(BSTR messageText);
```

## Parameters

*messageText*  
The message text.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::OnChatSessionEnd Event

---

Fired after a chat session has been closed.

## Syntax

```
HRESULT OnChatSessionEnded();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::OnChatSessionStart Event

---

Fired after a chat session has been initiated.

## Syntax

```
HRESULT OnChatSessionStarted();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# ISmartCodeVNCViewer::OnDisableRemoteInputEvent

---

Fired when ViewerX receives "Disable Remote Input" state from VNC server.

## Syntax

```
HRESULT OnDisableRemoteInputChanged(VARIANT_BOOL remoteInputEnabled)
```

## Parameters

*remoteInputEnabled*

Equals to VARIANT\_TRUE if "remote input" is enabled and VARIANT\_FALSE if "remote input" is disabled.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# ISmartCodeVNCViewer::OnMouseMove Event

---

Fired when the server-side cursor moves. For performance reasons, this event is not fired by default. Use [EnableMouseMoveEvent](#) property to enable this event.

## Syntax

```
HRESULT OnMouseMove(long x, long y);
```

## Parameters

*x*  
Specifies the x-coordinate of the cursor.

*y*  
Specifies the y-coordinate of the cursor.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# ISmartCodeVNCViewer::ScreenUpdated Event

---

Fired when a screen update has been received from a remote server. For performance reasons, this event is not fired by default. Use [EnableScreenUpdatedEvent](#) property to enable this event.

## Syntax

```
HRESULT ScreenUpdated();
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.



# ISmartCodeVNCViewer::ServerDimension Event

---

Fired when control has received a dimension of remote server screen.

## Syntax

```
HRESULT ServerDimension(long nWidth, long nHeight);
```

## Parameters

*nWidth*

**long** that specifies width of remote server screen.

*nHeight*

**long** that specifies height of remote server screen.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IScVxAdvancedSettings::EnableHotKeyCtrlA Property

---

This property sets/gets a value that indicates whether the CTRL+ALT+DEL(HOME) shortcut can be used during a remote session.

## Syntax

```
HRESULT IScVxAdvancedSettings::get_EnableHotKeyCtrlA  
HRESULT IScVxAdvancedSettings::put_EnableHotKeyCtrlA
```

## See Also

[IScVxAdvancedSettings::HotKeyCtrlAltDel](#)

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.

# IscVxAdvancedSettings::EnableScreenUpdate Property

---

Enables/Disables firing of [ScreenUpdated](#) event. For performance reasons, this property is set to False by default.

## Syntax

```
HRESULT IscVxAdvancedSettings::get_EnableScreenUpdatedEvent(VARIANT  
HRESULT IscVxAdvancedSettings::put_EnableScreenUpdatedEvent(VARIANT
```

## Parameters

*pbEnable*

Pointer to a variable of type **VARIANT\_BOOL** that receives ScreenUpdated event status.

*bEnable*

**VARIANT\_BOOL** that enables or disables ScreenUpdated event.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IScVxAdvancedSettings::EnableMouseMoveProperty

---

Enables/Disables firing of [OnMouseMoveEvent](#) event. For performance reasons, this property is set to False by default.

## Syntax

```
HRESULT IScVxAdvancedSettings::get_EnableServerMouseMoveEvent(VARI  
HRESULT IScVxAdvancedSettings::put_EnableServerMouseMoveEvent(VARI
```

## Parameters

*pbEnable*

Pointer to a variable of type **VARIANT\_BOOL** that receives EnableMouseMoveEvent event status.

*bEnable*

**VARIANT\_BOOL** that enables or disables EnableMouseMoveEvent event.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IScVxAdvancedSettings::HotKeyCtrlAltDel Property

---

Specifies the virtual-key code to add to CTRL+ALT to determine the hotkey replacement for sending CTRL+ALT+DEL command.

## Syntax

```
HRESULT IScVxAdvancedSettings::get_HotKeyCtrlAltDel(  
    HRESULT IScVxAdvancedSettings::put_HotKeyCtrlAltDel(  

```

## Property Value

The new virtual-key code. VK\_END is the default value.

## See Also

[IScVxAdvancedSettings::EnableHotKeyCtrlAltDel](#)

# IscVxCapabilities::Chat Property

---

Returns a value indicating whether the remote VNC server supports VNC chat. You should call this method only when connection has been established already.

## Syntax

```
HRESULT IScVxCapabilities::get_Chat(VARIANT_BOOL* pbSupport);
```

## Parameters

*pbSupport*

Pointer to a variable of type **VARIANT\_BOOL** that presents whether the VNC server supports VNC chat.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IscVxCapabilities::FileTransfer Property

---

Returns a value indicating whether the remote VNC server supports VNC file transfers. You should call this method only when connection has been established already.

## Syntax

```
HRESULT IScVxCapabilities::get_FileTransfer(VARIANT_BOOL* pbSupport)
```

## Parameters

*pbSupport*

Pointer to a variable of type **VARIANT\_BOOL** that presents whether the VNC server supports VNC file transfers.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IscVxCapabilities::EnableRemoteInput Function

---

Returns TRUE if the remote server supports enable/disable remote input feature. Supported by UltraVNC server only.

## Syntax

```
HRESULT IscVxCapabilities::get_EnableRemoteInput(VARIANT_BOOL* pbS
```

## Parameters

*pbSupport*

Pointer to a variable of type **VARIANT\_BOOL** that presents whether the VNC server supports mouse and keyboard input enabling/disabling.

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# IScVxCapabilities::RfbProtocolVersionMajor Property

---

Returns the major version number of the RFB protocol version used to communicate with the VNC server. You should call this method only when connection has been established already.

## Syntax

```
HRESULT IScVxCapabilities::get_RfbProtocolVersionMajor(int* pnMajor)
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# IScVxCapabilities::RfbProtocolVersionMinor Property

---

Returns the minor version number of the RFB protocol version used to communicate with the VNC server. You should call this method only when connection has been established already.

## Syntax

```
HRESULT IScVxCapabilities::get_RfbProtocolVersionMinor(int* pnMajc
```

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IscVxCapabilities::SelectSingleWindow Function

---

Returns TRUE if the remote server supports single window selection.  
Supported by UltraVNC server only.

## Syntax

```
HRESULT IscVxCapabilities::get_SelectSingleWindow(VARIANT_BOOL* pb
```

## Parameters

*pbSupport*

Pointer to a variable of type **VARIANT\_BOOL** that presents whether the VNC server supports single window selection.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# IscVxCapabilities::SwitchMultiMonitor Function

---

Returns TRUE if the remote server supports switching between monitors. Supported by UltraVNC server only.

## Syntax

```
HRESULT IscVxCapabilities::get_SwitchMultiMonitor(VARIANT_BOOL* pb
```

## Parameters

*pbSupport*

Pointer to a variable of type **VARIANT\_BOOL** that presents whether the VNC server supports monitors switching.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

---

# IScVxUltraSecurity\_MSRC4::KeyData Property

---

Sets UltraVNC MSRC4 DSM plug-in RC4 encryption key by passing a SAFEARRAY of bytes.

## Syntax

```
HRESULT IScVxUltraSecurity_MSRC4::put_KeyData(VARIANT arrDsmKeyData
```

## Parameters

*arrDsmKeyData*

VARIANT that holds SAFEARRAY of bytes with RC4 key file content.

## Remarks

In order for ViewerX to read RC4 key from a memory buffer, you must set IScVxUltraSecurity\_MSRC4::[KeyStorage](#) property value to [DKS\\_MEMORY](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IScVxUltraSecurity\_MSRC4::KeyDataAsHex Property

---

Sets MSRC4 RC4 key in form of HEX string. Each byte must be represented by two hexadecimal characters. If the byte value can be represented by a single hex character it must be padded with zero value. For example, 0F

## Syntax

```
HRESULT IScVxUltraSecurity_MSRC4::put_KeyDataAsHexStr(BSTR strDsmk
```

## Parameters

*strKeyPath*

BSTR that holds absolute path to encryption key file. The path can be an absolute path or a relative one.

## Remarks

In order for ViewerX to read RC4 key from a memory buffer, you must set IScVxUltraSecurity\_MSRC4::KeyStorage property value to [DKS\\_MEMORY](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.otherwise.

# IScVxUltraSecurity\_MSRC4::KeyFilePath Property

---

Gets or sets file path to UltraVNC MSRC4 DSM plug-in encryption key.

## Syntax

```
HRESULT IScVxUltraSecurity_MSRC4::get_KeyFilePath(BSTR* pstrKeyPat  
HRESULT IScVxUltraSecurity_MSRC4::put_KeyFilePath(BSTR strKeyPath)
```

## Parameters

*pstrKeyPath*

Pointer to a variable of type BSTR that receives path to the encryption key.

*strKeyPath*

BSTR that holds absolute path to encryption key file. The path can be an absolute path or a relative one.

## Remarks

In order for ViewerX to read RC4 key from an external file, you must set IScVxUltraSecurity\_MSRC4::[KeyStorage](#) property value to [DKS\\_FILE](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IscVxUltraSecurity\_MSRC4::KeyStorage Property

---

Specifies which key storage ViewerX must use to read RC4 key data.

## Syntax

```
HRESULT IscVxUltraSecurity_MSRC4::get_KeyStorage(DsmKeyStorage* pnStorage)  
HRESULT IscVxUltraSecurity_MSRC4::put_KeyStorage(DsmKeyStorage nStorage)
```

## Parameters

*pnStorage*

Pointer to a variable of type [DsmKeyStorage](#) that receives a key storage used to store RC4 key.

*nStorage*

[DsmKeyStorage](#) that specifies key storage used to store RC4 key.

## Remarks

Default value: DKS\_FILE

You can use one of the following properties to pass RC4 key to ViewerX: [KeyFilePath](#), [KeyData](#), or [KeyDataAsHexStr](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.



# IScVxUltraSecurity\_SecureVNC::PrivateKeyProperty

---

Sets UltraVNC SecureVNC DSM plug-in client authentication key by passing a SAFEARRAY of bytes.

## Syntax

```
HRESULT IScVxUltraSecurity_SecureVNC::put_PrivateKeyData(VARIANT a
```

## Parameters

*arrDsmKeyData*

VARIANT that holds SAFEARRAY of bytes with client authentication key file content.

## Remarks

In order for ViewerX to read SecureVNC client authentication key from a memory buffer, you must set IScVxUltraSecurity\_SecureVNC::[KeyStorage](#) property value to [DKS\\_MEMORY](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IScVxUltraSecurity\_SecureVNC::PrivateKeyI Property

---

Sets SecureVNC client key in form of HEX string. Each byte must be represented by two hexadecimal characters. If the byte value can be represented by a single hex character it must be padded with zero value. For example, 0F

## Syntax

```
HRESULT IScVxUltraSecurity_SecureVNC::put_PrivateKeyDataAsHexStr(E
```

## Parameters

*strKeyPath*

BSTR that holds absolute path to client authentication key file. The path can be an absolute path or a relative one.

## Remarks

In order for ViewerX to read SecureVNC client authentication key from a memory buffer, you must set IScVxUltraSecurity\_SecureVNC::[KeyStorage](#) property value to [DKS\\_MEMORY](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.otherwise.

# IScVxUltraSecurity\_SecureVNC::PrivateKeyProperty

---

Gets or sets file path to UltraVNC SecureVNC DSM plug-in encryption key (Viewer\_ClientAuth.pkey).

## Syntax

```
HRESULT IScVxUltraSecurity_SecureVNC::get_PrivateKeyFilePath(BSTR*  
HRESULT IScVxUltraSecurity_SecureVNC::put_PrivateKeyFilePath(BSTR
```

## Parameters

*pstrKeyPath*

Pointer to a variable of type BSTR that receives path to the client authentication key.

*strKeyPath*

BSTR that holds absolute path to client authentication key file. The path can be an absolute path or a relative one.

## Remarks

In order for ViewerX to read SecureVNC client authentication key from an external file, you must set IScVxUltraSecurity\_SecureVNC::[KeyStorage](#) property value to [DKS\\_FILE](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IScVxUltraSecurity\_SecureVNC::KeyStorage Property

---

Specifies which key storage ViewerX must use to read SecureVNC client authentication key data.

## Syntax

```
HRESULT IScVxUltraSecurity_SecureVNC::get_KeyStorage(DsmKeyStorage  
HRESULT IScVxUltraSecurity_SecureVNC::put_KeyStorage(DsmKeyStorage
```

## Parameters

*pnStorage*

Pointer to a variable of type [DsmKeyStorage](#) that receives a key storage used to store SecureVNC client authentication key.

*nStorage*

[DsmKeyStorage](#) that specifies key storage used to store client authentication key.

## Remarks

Default value: DKS\_FILE

You can use one of the following properties to pass SecureVNC client authentication key to ViewerX: [PrivateKeyFilePath](#), [PrivateKeyData](#), or [PrivateKeyDataAsHexStr](#).

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# IscVxUltraSecurity\_SecureVNC::Passphrase Property

---

Sets a passphrase to decrypt the private client key.

## Syntax

```
HRESULT IscVxUltraSecurity_SecureVNC::put_Passphrase(BSTR strPassp
```

## Parameters

*strPassphrase*

Passphrase of the private key file.

## Return Value

Returns S\_OK if successful, or an error value otherwise.

# DsmKeyStorage Enumeration

---

```
enum DsmKeyStorage
{
    DKS_FILE      = 0,
    DKS_MEMORY    = 1
};
```

---

© 2003-2015 [SmartCode Solutions](#). All rights reserved.