Citrix® ICA® Client Object API Specification Programmer’s Guide

Citrix Presentation Server Clients 10.x for Windows®

Citrix Presentation Server™ 4.5

Contents

Chapter 1 **Welcome**

Audience 11
Documentation Conventions 11
Getting More Information and Help 12
  Accessing Product Documentation 12
  Getting Service and Support 13

Chapter 2 **Introduction to the ICA Client Object**

What Is ICA? 16
What Are Citrix Presentation Server Clients? 16
What is the ICA Client Object? 16
ICA Client Object Requirements 17
ICA Client Object Environment 17
Embedding and Scripting Interfaces 19
  ActiveX 19
  Firefox Plug-ins 21

Chapter 3 **ICA Client Object Features**

ActiveX Control Support for Internet Explorer Security Zones 24
Automatic Application Resizing 26
  Function 26
  Methods, Properties, and Events 28
  Usage 30
Client Network Name and Address 30
  Function 30
  Methods 30
  Usage 31

Enabling or Disabling the Keyboard or Mouse 31
Function 31
Methods 31
Usage 32
Error Message Text From Error Code 32
Function 32
Methods 32
Usage 32
Firefox Plug-ins. 33
ICA Client Object Connection Performance Improvements. 33
ICA Client Object Global Log Off and Disconnect. 33
Features 34
Methods, Events, Properties, and Error Codes 36
Sample Code 38
Maximum Allowed Buffer Length. 38
Name Enumeration 38
Methods 39
When to Use Name Enumeration 39
Session Caching 39
Features 40
Methods, Properties, and Client Settings 44
When to Use Session Caching 45
Virtual Channel Support 46
Data Types 46
Methods, Properties, and Events. 46
Usage 47
Best Practices for Creating Virtual Channels 48

Chapter 4 **ICA Client Object APIs**

Property Interface 50
ClearProps 50
DeleteProp 51
DeletePropByIndex 52
GetPropCount 53
GetPropNameByIndex 54
GetPropValue 55
GetPropValueByIndex 56
ResetProps 57
SetProp. 58

Methods Interface 59
AboutBox. 59
AttachSession 60
CloseEnumHandle. 61
Connect 62
CreateChannels 63
DeleteWindow. 64
DetachSession 65
DisableKeyboardInput. 66
DisableMouseInput 67
DisableSizingBorder 68
Disconnect 69
DisconnectSessions 70
DisplayWindow. 71
DockWindow. 72
EnableKeyboardInput 73
EnableMouseInput. 74
EnableSizingBorder. 75
EnumerateApplications 76
EnumerateFarms 77
EnumerateServers 78
FocusWindow 79
FullScreenWindow 80
GetCachedSessionCount. 81
GetChannelData 82
GetChannelDataSize 83
GetChannelDataType 84
GetChannelCount 85
GetChannelFlags 86
GetChannelName 87
GetChannelNumber. 88
GetClientAddress 89
GetClientAddressCount. 90
GetErrorMessage. 91
GetClientErrorMessage. 92
GetInterfaceVersion 93
GetClientIdentification 94
GetClientNetworkName 95
GetEnumNameCount 96
GetEnumNameByIndex 97

GetGlobalChannelCount 98
GetGlobalChannelName 99
GetGlobalChannelNumber 100
GetLastClientError 101
GetLastError 102
GetMaxChannelCount. 103
GetMaxChannelWrite 104
GetMaxChannelRead 105
GetNotificationReason 106
GetScreenColorDepth 108
GetSessionCount 109
GetSessionCounter 110
GetSessionGroupCount. 111
GetSessionHandle 112
GetSessionHandleByIndex 113
GetScreenHeight 114
GetSessionString 115
GetScreenWidth 116
GetSessionColorDepth 117
GetSessionHeight 118
GetSessionWidth 119
GetWindowHeight. 120
GetWindowWidth 121
GetWindowXPosition 122
GetWindowYPosition 123
HideWindow 124
HideTitleBar 125
IsConnected 126
IsKeyboardInputEnabled. 127
IsMouseInputEnabled 128
IsPassThrough 129
IsSessionAttached 130
IsSessionDetached. 131
IsSessionRunning 132
IsWindowDocked 133
LoadIcaFile 134
Logoff 135
LogoffSessions 136
MaximizeWindow. 137
MinimizeWindow 138

NewWindow 139
PlaceWindowOnBottom 140
PlaceWindowOnTop 141
RestoreWindow 142
RunPublishedApplication 143
ScaleDialog 144
ScaleDisable 145
ScaleDown 146
ScaleEnable 147
ScalePercent 148
ScaleSize 149
ScaleToFit 150
ScaleUp 151
SetSessionGroupId 152
SendChannelData 153
SetChannelFlags 154
SetSessionEndAction 155
SetSessionId 156
SetWindowPosition 157
SetWindowSize 158
ShowTitleBar 159
Startup 160
SwitchSession 162
UndockWindow 163

Events Interface. 164
OnClick 166
OnConnect. 167
OnConnectFailed. 168
OnConnecting 169
OnDisconnect 170
OnDisconnectFailed 171
OnICAFile 172
OnICAFileFailed 173
OnLogon 174
OnPublishedApp 175
OnPublishedAppFailed 176
OnInitializing. 177
OnInitialProp 178
OnLogoffFailed. 179
OnChannelDataReceived 180
OnWindowSized 181
OnWindowMoved. 182
OnWindowCreated 183
OnWindowDestroyed 184
OnWindowDocked 185
OnWindowUndocked 186
OnWindowMinimized. 187
OnWindowMaximized 188
OnWindowRestored 189
OnWindowFullscreened 190
OnWindowHidden. 191
OnWindowDisplayed 192
OnWindowCloseRequest 193
OnDisconnectSessions 194
OnDisconnectSessionsFailed 195
OnLogoffSessions 196
OnLogoffSessionsFailed. 197
OnSessionSwitch. 198
OnSessionEventPending 199
OnSessionAttach 200
OnSessionDetach. 201

Chapter 5 ICA Client Object Properties
Accessing Properties 204
ICA Client Object Properties 204
ICA Browser Name Resolution 215
  ICA Browser–Specific Properties 215
  ICA Browser Address Grid 217
About Passwords 218
About Scaling 218

Chapter 6 Example Scripts
Embedding 222
  ActiveX 222
  Firefox Plug-in 223
Scripting 223
  ActiveX 224
  Firefox Plug-in 226
Logon Methods 227
  ActiveX 227
  Firefox Plug-in 228
Custom User Interfaces 229
  ActiveX 229
  Firefox Plug-in 230
Mouse Click 231
  ActiveX 231
  Firefox Plug-in 233
Scaling Methods 235
  ScaleEnable and ScaleDisable 235
  ScaleUp and ScaleDown 237
  ScalePercent 239
  ScaleToFit 241
  ScaleDialog 243
Scaling Properties 245
  Example 1 245
  Example 2 246
  Example 3 248
  Example 4 249
Client-Side HTML/VBScripting Using ActiveX 250
Client-Side HTML/VBScripting Using the Firefox Plug-In 253
Session Caching 256

Name Enumeration 261
Virtual Channel Support 264
Automatic Application Resizing 270
Client Network Name and Address 278
Error Message Text from Error Codes 280
Global Logoff and Disconnect 282

Chapter 7 **Error Codes**

ICA Client Object Error Codes 298
Client Error Codes 303

End User License Agreement 311

Index 317

Welcome

Welcome to Version 10.x of the ICA Client Object (ICO) specification for the Independent Computing Architecture (ICA) Client for Windows operating systems.

You should have the Citrix Presentation Server Independent Client Architecture (ICA) Clients for Windows (Version 10.100 or later) installed and working on the client device and be familiar with basic embedding and scripting tools and techniques.

Audience

This document is intended for Citrix Presentation Server administrators, Web masters, Independent Software Vendors (ISVs), and power users of the Presentation Server Client who are looking to integrate the client with environments and custom applications that support object embedding.

Documentation Conventions

Citrix documentation uses the following typographic conventions for menus, commands, keyboard keys, and items in the program interface:

Convention	**Meaning**
Boldface	Commands, names of interface items such as text boxes, option buttons, and user input.
Italics	Placeholders for information or parameters that you provide. For example, *filename* in a procedure means you type the actual name of a file. Italics also are used for new terms and the titles of books.
%SystemRoot%	The Windows system directory, which can be WTSRV, WINNT, WINDOWS, or other name you specify when you install Windows.
`Monospace`	Text displayed in a text file.

Convention	Meaning
{ braces }	A series of items, one of which is required in command statements. For example, **{ yes | no }** means you must type **yes** or **no**. Do not type the braces themselves.
[brackets]	Optional items in command statements. For example, **[/ping]** means that you can type **/ping** with the command. Do not type the brackets themselves.
| (vertical bar)	A separator between items in braces or brackets in command statements. For example, **{ /hold | /release | /delete }** means you type **/hold** or **/release** or **/delete**.
… (ellipsis)	You can repeat the previous item or items in command statements. For example, **/route**:*devicename*[,…] means you can type additional *devicenames* separated by commas.

Getting More Information and Help

This section describes how to get more information about Citrix products and how to contact Citrix.

Accessing Product Documentation

Citrix product documentation includes PDF guides, online documentation, known issues information, integrated on-screen assistance, and application help.

- Online documentation is provided as Adobe Portable Document Format (PDF) files. To view, search, and print the PDF documentation, you need Adobe Reader (supported versions include 5.0.5 with Search, Version 6, 7, or 8).
- Be sure to read the Readme file before installation or when troubleshooting. This file contains important information that includes last-minute documentation updates and corrections.
- The *Citrix Presentation Server Administrator's Guide* provides conceptual information and procedures for system administrators who plan, design, pilot, deploy, and administer Presentation Server.
- The *Clients for Windows Administrator's Guide* provides information and procedures for installing, configuring, deploying, and maintaining the Citrix Presentation Server Clients for Windows.

All Microsoft Terminal Server compatibility guidelines can be applied to Presentation Server. For Terminal Server compatibility information, see the following Microsoft resources:

- The Microsoft Web site, http://www.microsoft.com
- Microsoft TechNet

Important To provide feedback about the documentation, go to http://support.citrix.com/docs/. To access the feedback form, click the **Submit Documentation Feedback** link.

Getting Service and Support

Citrix provides technical support primarily through the Citrix Solutions Network (CSN). Our CSN partners are trained and authorized to provide a high level of support to our customers. Contact your supplier for first-line support or check for your nearest CSN partner at http://www.citrix.com/support/.

In addition to the CSN channel program, Citrix offers a variety of self-service, Web-based technical support tools from its Knowledge Center at http://support.citrix.com/. Knowledge Center features include:

- A knowledge base containing thousands of technical solutions to support your Citrix environment
- An online product documentation library
- Interactive support forums for every Citrix product
- Access to the latest hotfixes and service packs
- Security bulletins
- Online problem reporting and tracking (for customers with valid support contracts)

Another source of support, Citrix Preferred Support Services, provides a range of options that allows you to customize the level and type of support for your organization’s Citrix products.

Subscription Advantage

Subscription Advantage gives you an easy way to stay current with the latest server-based software functionality and information. Not only do you get automatic delivery of feature releases, software upgrades, enhancements, and maintenance releases that become available during the term of your subscription, you also get priority access to important Citrix technology information.

You can find more information on the Citrix Web site at http://www.citrix.com/services/ (select Subscription Advantage). You can also contact your Citrix sales representative or a member of the Citrix Solutions Network for more information.

Customizing Citrix Products

The Citrix Developer Center at http://support.citrix.com/developers is the home for all technical resources and discussions involving the use of Citrix Software Development Kits (SDKs). You can find access to SDKs, sample code and scripts, extensions and plug-ins, and SDK documentation. Also included are the Citrix Developer Network forums, where technical discussions take place around each of the Citrix SDKs.

Education and Training

Citrix offers a variety of instructor-led training and Web-based training solutions. Instructor-led courses are offered through Citrix Authorized Learning Centers (CALCs). CALCs provide high-quality classroom learning using professional courseware developed by Citrix. Many of these courses lead to certification.

Web-based training courses are available through CALCs, resellers, and from the Citrix Web site.

Information about programs and courseware for Citrix training and certification is available from http://www.citrix.com/edu/.

Introduction to the ICA Client Object

This chapter contains an overview of Independent Client Architecture (ICA) and describes system requirements for the ICA Client Object (ICO) and the environment in which it operates. This chapter includes the following topics:

- "What Is ICA?"
- "What Are Citrix Presentation Server Clients?"
- "What is the ICA Client Object?"
- "ICA Client Object Requirements"
- "ICA Client Object Environment"
- "Embedding and Scripting Interfaces"

Understanding these topics provides you with a context for the remaining chapters of this book.

What Is ICA?

ICA is the foundation of Citrix server-based computing with Citrix Presentation Server Client software. The ICA protocol sends screen data from a computer running Presentation Server to the client users and returns the users' input to the application on the server.

When a user types on the keyboard or moves and clicks the mouse, the client sends this data to the application on the server. The Citrix ICA protocol provides advanced capabilities and enhanced performance with Windows Terminal Services. ICA delivers high performance on high- and low-bandwidth connections. It requires minimal client workstation capabilities and includes error detection and recovery, encryption, and data compression.

What Are Citrix Presentation Server Clients?

The Citrix Presentation Server Clients (formerly known as the ICA clients) are the software components of Presentation Server that execute on the client device. The clients allow the user to establish an ICA session with a computer running Presentation Server. With this session the user can access server-based applications that appear to run locally on the client device but actually execute on the server.

Client software extends the reach of Windows and UNIX-based applications to virtually any client platform or device. These include information appliances, handheld computers, network computers, X-devices, Windows-based terminals, and PCs (Pentium, 486, and 386). Clients are available for all types of Windows operating systems including Windows CE (Windows-based terminals and Handheld PCs), Windows NT Workstation, Windows 95/98, Windows Me, Windows 2000 Professional, Windows XP, Windows 2003, Windows Vista, Windows for Workgroups, and Windows 3.x.

Web Clients (ActiveX control and Firefox Plug-in) allow Citrix administrators to deploy applications through their intranets and ASP customers to deploy applications through public networks, such as the Internet.

What is the ICA Client Object?

The ICA Client Object specification provides an Application Programming Interface (API) for Presentation Server Clients for Windows. Use this API to control the appearance and behavior of Presentation Server Clients. You can embed the ICA Client Object in custom software applications to enable users to launch ICA sessions with a click of a button, while shielding them from the mechanics of accessing a server farm.

Any application that supports object embedding can interface with the client and pass instructions to it. The ICA Client Object is the framework that exposes the functionality of the client to other applications or objects.

With the ICA Client Object API, you can:

- Embed the client in commercially available desktop applications that support object embedding, such as Internet Explorer, Firefox, and the Microsoft Office suite
- Seamlessly embed and integrate ICA functionality into third-party applications

Use the ICA Client Object API within scripting environments to integrate and manipulate the appearance and behavior of the client.

ICA Client Object Requirements

The following environments are required.

Windows 32 Client Requirements

The Citrix Presentation Server Clients for Windows Version 10.100 or later is required.

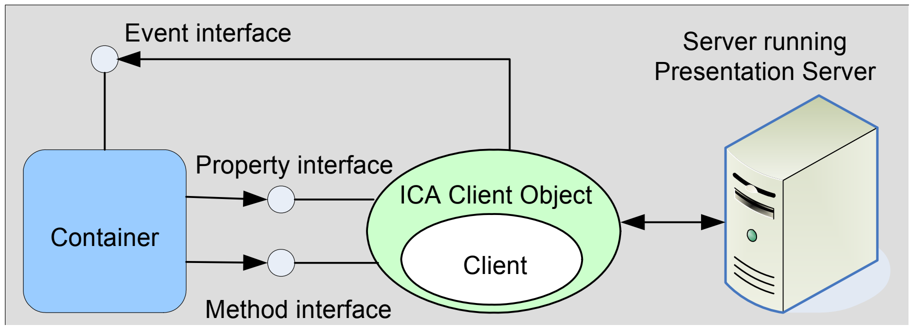
Server Requirements

One of the following:

- Citrix Presentation Server 4.5
- Citrix MetaFrame Presentation Server 3.0 or 4.0
- MetaFrame XP 1.0, 2.0, or 3.0
- MetaFrame 1.8 for Windows 2000
- MetaFrame 1.8 for Windows NT 4.0, Terminal Server Edition

ICA Client Object Environment

The ICA Client Object is the framework that exposes the functionality of the client to third party applications. The framework is based on an embedding model, where the ICA Client Object is contained within a host application called a *container*.



This diagram outlines the interfaces between the ICA Client Object and the container.

The ICA Client Object framework includes the following APIs:

- Method interface
- Property interface
- Event interface

The diagram above outlines the interfaces between the ICA Client Object and the container, where:

- A *container* is an instance of a third party container application. A container application can be a Web browser (Internet Explorer, Firefox), a Visual Basic application, a Delphi application, a Microsoft Foundation Class (MFC) application, or any application that supports the embedding technologies provided by the ICA Client Object.
- The *ICA Client Object* is an interface wrapper around the client. Each instance of the client is managed by an instance of the ICA Client Object. The ICA Client Object provides public interfaces to the client using standard technologies such as ActiveX and LiveConnect.
- The *Method Interface* is used by the container to control and interact with the ICA Client Object.
- The *Property Interface* is used by the container to access properties of the ICA Client Object.
- The *Event Interface* is used by the ICA Client Object to send events to the container.

Embedding and Scripting Interfaces

The ICA Client Object provides a set of standard scripting interfaces or APIs that allow you to interface with the client software and control it from third-party applications.

When an object is embedded in a container, you can control it with scripts. You can use scripting tools such as JavaScript and VBScript to control the object through the scripting interfaces. For example, in Word and Excel, you use VBScript; for Internet Explorer, you use VBScript and JavaScript; while for the Firefox browser, you use JavaScript.

ActiveX controls and Firefox plug-ins are two of the enabling technologies that allow you to embed objects within container applications such as Internet Explorer, Firefox, Microsoft Word, and Excel. The ICA Client Object implementation is based on these two embedding technologies.

ActiveX

ActiveX is based on the Component Object Model (COM) architecture. ActiveX controls are supported by the Internet Explorer Web browser environment and by any container application conforming to the COM interface.

Embedding

ActiveX controls can be embedded in a Web browser (Internet Explorer) through the OBJECT tag. For other types of container applications, ActiveX controls are embedded through the standard COM interface.

In a Web browser environment (Internet Explorer), an ActiveX control is embedded in a Web page as follows:

1. The ActiveX control must be installed and registered with the operating system. If the control is not already installed, use the OBJECT tag to specify a URL from which the ActiveX control can be downloaded and installed automatically.
2. When the ActiveX control is installed and registered, the Web browser creates an instance of the object (the client).
3. The ActiveX control receives the download data from the browser (the ICA file) and then starts the connection to the server. In addition to the downloaded ICA file, parameters can be specified through PARAM tags.

Properties

The ActiveX embedding model permits access to the public properties of an object. In a Web browser environment, properties of an object can be set through PARAM tags at load time, and can be read through the scripting interface. For other types of container applications, properties are accessible through the COM interface.

In addition to the PARAM tag interface, the ICA Client Object provides the property interface that permits the Web browser to enumerate and access properties through scripting. These methods are in the general form of `GetPropCount()`, `SetProp(propertyName, value)`, `GetPropValue(propertyName, value)`, and so on. Other types of container applications can access these interfaces through the COM interface. See "Property Interface" on page 50 for a complete list of property interfaces available for the ICA Client Object.

Methods

The ActiveX embedding model permits access to public methods of an object. For a Web browser, the object methods are accessed through supported scripting languages. For other types of container applications, the object methods are accessed through the traditional COM interface.

The ActiveX implementation of the ICA Client Object publishes a set of methods. These methods are in the general form of *DoAction(optionalParam)*. See "Methods Interface" on page 59 for a list of published methods.

Events

The ActiveX embedding model provides a standard interface through which events are dispatched to the container. In this interface model, the ActiveX control (the source of the events) defines a set of supported events, while the container defines a handler for each of the events in which it is interested. Event handlers are referred to as *event sinks*.

The ActiveX implementation of the ICA Client Object supports this events notification interface. The container provides the necessary event sink interface for each of the events it wants to receive. See "Events Interface" on page 164 for a list of available events.

Summary

The ActiveX implementation of the ICA Client Object supports embedding and scripting on any platform where COM architecture is implemented.

Embedding of the ActiveX control is not limited to Internet Explorer; embedding is supported by any application conforming to the COM interface.

Firefox Plug-ins

Plug-ins are a Firefox component technology. Firefox plug-ins support embedding and scripting. You can use the Firefox plug-in API to develop components that extend the capabilities of the Firefox browser.

Important The Firefox plug-in is modified to expose only those methods and properties available to the ICA Client Object ActiveX control in the Internet Explorer Internet security zone.

Installing

You can install Firefox plug-ins either per computer or per user, depending on the installation package used. The icapkg32.msi installs on the per-computer Firefox installation; the icaweb32.msi installs on the per-user Firefox installation. To install the Firefox plug-ins, install Firefox before running the client .msi packages.

Important If you are using Vista, be aware that Firefox does not support per-user installation on Vista platforms; therefore, you can install the Firefox ICA Client Object plug-ins only on an existing computer-wide Firefox installation (icapkg32.msi).

Embedding

A plug-in is embedded in a Web page as follows:

1. The plug-in must be present in the \Plugins directory of the Firefox Web browser. If the plug-in is not found, the user is prompted to download the plug-in from some location on the Web.
2. The plug-in is loaded using a well-defined API. The page that triggers the plug-in to load has either an EMBED or OBJECT tag with a MIME type supported by the plug-in.
3. The plug-in receives the downloaded data from the browser, then initiates the connection to the server. Parameters can be set using HTML tags.

Properties

Using LiveConnect for Firefox, you can set properties that are contained in the plug-in. From the script point of view, the plug-in is a Java object with methods and properties. The plug-in uses the LiveConnect environment to publish these scripting interfaces. If the Firefox Web browser does not support Java because it is disabled or is not supported, it is not possible to support calling methods in the plug-in.

The Firefox plug-in implementation of the ICA Client Object uses the LiveConnect environment to support methods, properties, and events.

The properties that can be set belong to several categories. For example, there are properties for modifying the behavior of the plug-in, properties for changing the connection information, and properties for retrieving information from the server. Scripts alter settings in different aspects of the ICA Client Object using properties. See "Property Interface" on page 50 for a complete list of property interfaces available for the ICA Client Object.

Methods

Methods are also dependent on LiveConnect support. The set of published methods that can be called is the same as for the ActiveX implementation. See "Methods Interface" on page 59 for a list of published methods.

Events

Events are intended to notify the container (Firefox Web browser) of any changes in the state of the ICA Client Object. See "Events Interface" on page 164 for a list of available events.

Summary

Firefox Web browsers provide support for plug-ins. Firefox plug-ins cannot be embedded in any other application. The Firefox plug-in requires LiveConnect to support scripting.

ICA Client Object Features

This chapter describes some of the more recent features and functionality added to the ICA Client Object (ICO) specification. The table lists the features and functionality added to the ICA Client Object and the versions in which they were added.

Feature	**ICA Client Object version in which it first appeared**
ActiveX Control Support for Internet Explorer Security Zones	2.3
Automatic Application Resizing	2.2
Client Network Name and Address	2.2
Enabling or Disabling the Keyboard or Mouse	2.2
Error Message Text From Error Code	2.2
Firefox Plug-ins	10.x
ICA Client Object Connection Performance Improvements	2.3
ICA Client Object Global Log Off and Disconnect	2.3
Maximum Allowed Buffer Length	10.x
Name Enumeration	2.2
Session Caching	2.2
Virtual Channel Support	2.2

ActiveX Control Support for Internet Explorer Security Zones

Citrix digitally signs the Citrix Presentation Server ActiveX control, rendering it safe for scripting and initialization.

In earlier releases, some ICA Client Object methods, properties, and events were potentially unsafe for scripting because they could allow untrusted hosts (in the Restricted Sites and Internet zones) to access information. Industry standards for safe scripting require that a script from an untrusted host be prevented from using ICA Client Object interfaces to do any of the following:

- Read properties that were not previously set.
- Read or write any security-related or otherwise sensitive client configuration information.
- Exert any influence on the client's execution environment (for example, to control which executable code, including DLLs, is loaded by the client).
- Create full-screen, hidden, or offscreen windows.
- Influence the behavior of other client-side applications (for example, to configure hotkeys to cause user input to have an unexpected effect on another client-side application).
- Use ICA Client Object interfaces to perform any action that Microsoft Internet Explorer security policy settings prevent. For example, a script from an untrusted host must not be able to download any file to the client device if **File Download** permission is disabled in the security zone from which the script originated.

The safe scripting enhancements prevents scripts from using ICA Client Object interfaces except when they are launched from the local computer, trusted sites, or intranet zones.

In addition, permitted initialization parameters are mapped to the same security zones so that the client is safe for initialization and scripting.

Internet Explorer Security Zones	**Scripting of Interfaces Permitted?**	**Initialization Permitted?**
Local Computer	Yes	Yes
Local Intranet	Yes	Yes

Internet Explorer Security Zones	**Scripting of Interfaces Permitted?**	**Initialization Permitted?**
Trusted Sites	Yes	Yes
Restricted Sites	No	No
Internet Restricted sites and Internet zones can download only HTTP and HTTPS type ICA files.	No	Initialization is permitted for the following parameters and events: Parameters - Start, Border, Width, Height, SRC, CacheICAFile, SessionId, SessionCacheEnable, AutoAppResize, and ICAFile. Events - OnConnect, OnConnectFailed, OnDisconnect, OnIcaFile, OnIcaFileFailed.

Important Users who already use these constrained ActiveX control facilities for sites in the Internet zone must add the sites to the trusted sites zone. You must not use ActiveX controls in the restricted sites zone and they are disabled by default in that zone in Internet Explorer.

To allow backward compatibility, an additional setting that disables the above checks, IgnoreZones, is included in Webica.ini. To disable the checks, add the following entry in the Webica.ini file:

```
[Zones]

IgnoreZones=TRUE
```

By default, this setting is not present in the .ini file. The Webica.ini file resides in the user's Windows directory.

Important Ensure that you correctly set up the security zones in Internet Explorer if you are using the ICA Client Object control. Scripting functionality is disabled if security zones are set up incorrectly.

Automatic Application Resizing

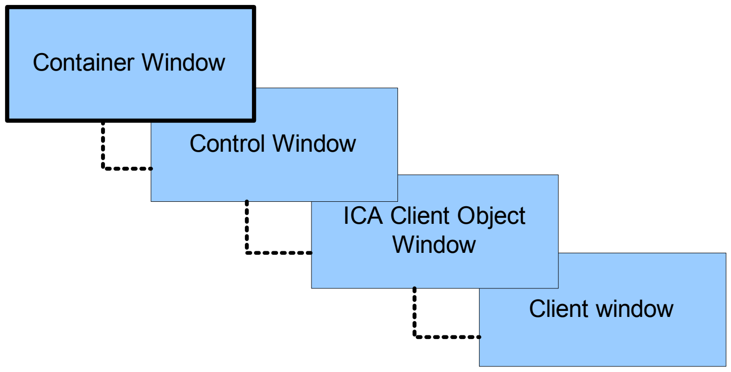
This functionality allows you to resize a server application window within the ICA Client Object container. In the past, when the ICA Client Object container was resized, the server application window size remained intact. With automatic application resizing, you can resize the server application window to the required dimensions.

Function

Automatic application resizing gives you the ability to force the server application to resize based on requests from the client. The server is notified whenever the size of the ICA Client Object container changes. When notified, the server resizes the published application or resource window accordingly.

The automatic application resizing feature provides the necessary interfaces that allow ICA Client Object windows to be resized, moved, and modified. Windows can be undocked and placed on the desktop, allowing minimizing, maximizing, restoring, and full screen window operations. An undocked window can be redocked into the Web page just as easily.

Automatic application resizing also provides the ability to set the Z ordering of ICA Client Object windows. The Z ordering hierarchy of ICA Client Object windows is quite strict; that is, the container window is the parent of the control window; which is the parent of the ICA Client Object window; which is the parent of the client window.



This diagram illustrates the Z ordering hierarchy of the ICA Client Object windows.

Use the AutoAppResize property to enable automatic application resizing. The default value of AutoAppResize is False.

ICA Client Object Window Types

This table contains the ICA Client Object window types, the identifier, and value.

Window Type	Identifier	Value
ICA Client Object window	ICO_WINDOW_TYPE_ICO	0
Control window	ICO_WINDOW_TYPE_CONTROL	1
Client window	ICO_WINDOW_TYPE_CLIENT	2
Container window	ICO_WINDOW_TYPE_CONTAINER	3

ICA Client Object Window Position Flags

This table contains the ICA Client Object window position flag types, identifiers, and values.

Position Type	Identifier	Value
Client area origin point	ICO_WINDOW_FLAG_POS_CLIENT_AREA_ORIGIN	0
Window area origin point	ICO_WINDOW_FLAG_POS_WINDOW_AREA_ORIGIN	1
Parent relative position	ICO_WINDOW_FLAG_POS_PARENT_RELATIVE	0
Screen relative position	ICO_WINDOW_FLAG_POS_SCREEN_RELATIVE	2

ICA Client Object Window Area Flags

This table contains the ICA Client Object window area flag types, identifiers, and values.

Area Type	Identifier	Value
Client area	ICO_WINDOW_FLAG_SIZE_CLIENT_AREA	0
Window area	ICO_WINDOW_FLAG_SIZE_WINDOW_AREA	1
Create Docked window	ICO_WINDOW_FLAG_NEW_DOCKED	0
Create Undocked window	ICO_WINDOW_FLAG_NEW_UNDOCKED	1

Methods, Properties, and Events

Use the following methods, properties, and events to perform automatic application resize operations.

Methods

These methods apply to automatic application resizing.

Name	Description
GetWindowWidth	Get the width of the specified window type based on area flags.
GetWindowHeight	Get the height of the specified window type based on area flags.
GetWindowXPosition	Get the X position of the specified window type based on position flags.
GetWindowYPosition	Get the Y position of the specified window type based on position flags.
SetWindowSize	Set the window size for the window specified based on area flags.
SetWindowPosition	Set the window position for the window specified based on position flags.
DisplayWindow	Display the specified window.
PlaceWindowOnTop	Place the ICA Client Object window on top of other windows.
PlaceWindowOnBottom	Place the ICA Client Object window at the very bottom of other windows.
ShowTitleBar	Show the title bar on the undocked ICA Client Object window.
HideTitleBar	Hide the title bar on the undocked ICA Client Object window.
EnableSizingBorder	Enable the sizing border on the undocked ICA Client Object window.
DisableSizingBorder	Disable the sizing border on the undocked ICA Client Object window.
FocusWindow	Give the ICA Client Object window the keyboard focus.
UndockWindow	Undock the ICA Client Object window from the control window.
DockWindow	Dock the ICA Client Object window in the control window.
MinimizeWindow	Minimize the ICA Client Object window.
MaximizeWindow	Maximize the ICA Client Object window.
RestoreWindow	Restore the ICA Client Object window to the original size.
FullScreenWindow	Make the ICA Client Object window a full screen window.
IsWindowDocked	Determine if a window is currently docked.

Name	Description
GetSessionWidth	Get the width of the current session in pixels.
GetSessionHeight	Get the height of the current session in pixels.
GetSessionColorDepth	Get the color depth of the current session in bits per pixel.
GetScreenWidth	Get the width of the screen in pixels.
GetScreenHeight	Get the height of the screen in pixels.
GetScreenColorDepth	Get the color depth for the screen. The number returned is bits per pixel for color.
NewWindow	Create a new ICA Client Object window (only if the ICA Client Object window no longer exists).
DeleteWindow	Delete the ICA Client Object window.

Properties

These properties apply to automatic application resizing.

Name	Description
AutoScale	Automatically scale the client session.
AutoAppResize	Automatically resize the client application to fit the ICA Client Object window.

Events

These events apply to automatic application resizing.

Event #	Handler	Description
16	OnWindowSized	Window size is changed.
17	OnWindowMoved	Window position is changed.
18	OnWindowCreated	Window is created.
19	OnWindowDestroyed	Window is destroyed.
20	OnWindowDocked	Window is docked.
21	OnWindowUndocked	Window is undocked.
22	OnWindowMinimized	Window is minimized.
23	OnWindowMaximized	Window is maximized.
24	OnWindowRestored	Window is restored.
25	OnWindowFullscreened	Window is set to full screen.

Event #	Handler	Description
26	OnWindowHidden	Window is hidden.
27	OnWindowDisplayed	Window is displayed.
28	OnWindowCloseRequest	Window is being requested to close.

For descriptions of the above methods, properties, and events, see "ICA Client Object APIs" on page 49. For code examples incorporating automatic application resizing functionality, see "Automatic Application Resizing" on page 270.

Usage

Automatic application resizing is particularly useful in a Web portal environment. You can now embed applications within a portal and resize them transparently to fit into a portal window of any size.

The ability to undock windows also means it is not necessary to force a session to be either embedded or launched. You can dock or undock the window dynamically through scripting. The interfaces provided with this feature enable you to do powerful, dynamic, and transparent operations with ICA Client Object windows.

Client Network Name and Address

This feature allows the network name and address to be retrieved from the ICA Client Object, enabling you to uniquely identify the client.

Function

This feature uses the local computer network name and TCP/IP stack to determine the information requested. In most cases, the network name corresponds to the assigned name of the client during network configuration. TCP/IP addresses are assigned from the Dynamic Host Configuration Protocol (DHCP) or locally configured settings.

Methods

The following methods apply to the client network name and address operations.

Name	Description
GetClientNetworkName	Gets the client's network name.
GetClientAddressCount	Gets the number of TCP/IP network addresses available.
GetClientAddress	Gets the TCP/IP network address by index.

For descriptions of the above methods, see “ICA Client Object APIs” on page 49. For code examples incorporating these methods, see “Client Network Name and Address” on page 278.

Usage

Client network name and address functionality is primarily used to track the IP address of the client device. The IP address is then used to access published application resources or to determine what client name should be used when logging on to a server.

Enabling or Disabling the Keyboard or Mouse

This feature provides the ability to turn keyboard and mouse input On or Off. It is useful when monitoring sessions or when multiple session windows are contained on a single page.

It enables primitive keyboard and mouse locking. Input is not processed after a disable until the corresponding enable is called.

Function

This feature provides the necessary switches required to stop the server from processing any keyboard or mouse input from the client.

Keyboard and mouse can be disabled individually. The current value of the setting, whether keyboard and mouse input is enabled or disabled, can also be determined.

Methods

The following methods apply to the enable and disable keyboard and mouse operations.

Name	Description
DisableKeyboardInput	Disables keyboard input for the session.
DisableMouseInput	Disables mouse input for the session.
EnableKeyboardInput	Enables keyboard input for the session.
EnableMouseInput	Enables mouse input for the session.
IsKeyboardInputEnabled	Determines if keyboard input is enabled.
IsMouseInputEnabled	Determines if mouse input is enabled.

For descriptions of the above methods, see "ICA Client Object APIs" on page 49. For code examples incorporating these methods, see "Automatic Application Resizing" on page 270.

Usage

This feature can be useful when you need to monitor active sessions in a portal environment. For instance, you can disable all interaction with a particular session so that user input cannot alter the state of that session.

Error Message Text From Error Code

This feature makes it possible to extract error message text for a particular error code.

Function

Given an error code from the ICA Client Object or the client, these methods return the corresponding error message text. The error messages are built into the ICA Client Object and are used for normal error processing; however, the container can use these messages when required.

Methods

The following methods apply to the error message text from error code operations

Name	**Description**
GetErrorMessage	Gets the ICA Client Object error message for error number.
GetClientErrorMessage	Gets the client error message for error code.

For descriptions of the above methods, see "ICA Client Object APIs" on page 49. For code examples incorporating these methods, see "Error Message Text from Error Codes" on page 280.

Usage

This feature makes it easier to relate error codes to their respective error messages and enables you to generate more informative error messages.

Firefox Plug-ins

Firefox plug-ins support embedding and scripting. You can use the Firefox plug-in API to develop components that extend the capabilities of the Firefox browser.

Important The Firefox plug-in is modified to expose only those methods and properties available to the ICA Client Object ActiveX control in the Internet Explorer Internet security zone.

ICA Client Object Connection Performance Improvements

Connection handling in a Web-based environment is significantly improved, permitting faster execution of connection requests. These are client-side enhancements; the modules affected are Wfica.ocx (the ActiveX control) and Wfica32.exe (the client engine).

Enhancement	**Description**
Connect to Server	Use of a nonblocking socket facilitates shorter connection times when connecting to multiple servers.
INI File Processing	Increased efficiencies in INI file processing due to more intelligent configuration data collection and reassembly. Other improvements include use of a memory image file for INI file processing.
Connect Thread	All connect processing occurs using a new thread so that the main thread is not hampered or blocked.

The above improvements significantly enhance the user experience when multiple ICA Client Object sessions are started from a Web page.

ICA Client Object Global Log Off and Disconnect

Global log off and disconnect functionality in a Web environment is used when users are connected to the secure network. When a user logs off, all sessions associated with that user are either logged off or disconnected automatically. Sessions that would normally be disconnected can now be logged off, which results in reducing the session load on servers.

These are client-side enhancements; the modules affected are Wfica.ocx (the ActiveX control) and Wfica32.exe (the client engine).

Enhancement	Description
Log off and Disconnect sessions	Logs off or disconnects a group of sessions.
SessionGroupId support	Facilitates grouping of individual sessions so that sessions can be disconnected or logged off in groups.
Launching IPC support	Applications are launched using IPC instead of directly launching the WFICA32 executable.This permits better control of launched sessions and facilitates global disconnect or log off.
Switching between launched sessions	Allows users to switch among multiple sessions launched through ICO.
Session handle support	Enables selecting or referring to other sessions using session handles.
Error codes	New error codes applicable to Global Log Off and Disconnect events are added.

Features

The following sections describe the individual global log off and disconnect operations.

Disconnect and Log off Multiple Sessions

Use the DisconnectSessions and LogoffSessions APIs to disconnect or log off multiple sessions. Any ICA Client Object object can call these interfaces.

DisconnectSessions and LogoffSessions are asynchronous by nature. A successful call to these APIs does not mean that all of the specified sessions are successfully logged off or disconnected.

To determine if the methods are successful, events are triggered to show the status of a disconnect or log off command. OnDisconnectSessions and OnLogoffSessions are triggered if successful; OnDisconnectSessionsFailed and OnLogoffSessionsFailed are triggered on failure. If the command fails, GetSessionGroupCount can be called to determine how many sessions remain active within the specified group.

The SessionExitTimeout property specifies the interval to wait before determining if a request failed. The default value of SessionExitTimeout is 60 seconds. If sessions within the specified group remain active after the interval specified by SessionExitTimeout, an event indicating that the command did not complete is triggered.

Group Sessions Using SessionGroupId

Use the SetSessionGroupId interface to specify a session group ID for an active ICA Client Object session. Sessions launched outside the access center, for example, through Program Neighborhood do not have a group ID and are not affected by the ICA Client Object.

The default value of SessionGroupId is **ICOGroup** and ICA Client Object sessions for which you do not specifically set SessionGroupId are assigned **ICOGroup** as the session group ID. You can set SessionGroupId prior to establishing a connection through a property; however, if the connection is already established, use the SetSessionGroupId method to assign a group ID to a session.

This guarantees that the engine has the correct SessionGroupId setting. DisconnectSessions and LogoffSessions internally enumerate a list of engines and issue the IPC commands for log off and disconnect to those sessions with the matching SessionGroupId. SessionGroupId is a write-only property and must not exceed a maximum length of 240 characters.

A built-in security measure ensures that you cannot retrieve the SessionGroupId value for an existing session. This prevents malicious scripts from obtaining the SessionGroupId and forcing the group sessions to log off or disconnect.

Launching Sessions Using Inter Process Communication

The implementation of Inter Process Communication (IPC) support for ICA sessions launched through the ICA Client Object means that scripts can now issue events and run-time methods to active ICA sessions. IPC also provides the ability to communicate simultaneously with multiple launched sessions. IPC support is necessary for global log off and disconnect functionality.

To enable or disable IPC support, use the property IPCLaunch. Note that when you disable IPCLaunch, the ICA Connection Center functionality becomes available. This might be required for users of Firefox or for users who want to use the ICA Connection Center.

Switching between Launched Sessions

To enable the ICA Client Object to switch between launched sessions, use GetSessionHandle to obtain the current session handle. This handle can be stored and used later if the launched session is still running. Use SetSessionHandle to switch session focus to a different launched session. To determine how many launched sessions are outstanding, use GetSessionCount. When a count of launched sessions is obtained, you can loop through the different sessions and get their handles using GetSessionHandleByIndex.

Only one session at a time can be the actively selected session inside the ICA Client Object. This includes launched sessions. To receive events from a different launched session, switch to that session. This is also true for all commands (for example, log off or disconnect). If a session is not currently selected, the events are stored and released when the session is reselected. To obtain a list of outstanding events for an inactive session, use OnSessionSwitch and OnSessionEventPending.

Improved Session Handling

An ICA Client Object session is represented by its session handle. The session handle is relevant to a specific ICA Client Object object only and is matched with a launched session using an internal linked list.

Session handles are specific to an ICA Client Object object; they cannot be used to access sessions launched from other ICA Client Object objects. This is an effective security mechanism to protect sessions from malicious access. The only session handles available to a particular ICA Client Object are the ones that it launched.

An embedded session is also assigned a handle. A session handle is invalid when set to zero. This enables the session handling the API to return zero as an error context for the session handle.

Methods, Events, Properties, and Error Codes

Use the following methods, properties, events, and error codes for the global log off and disconnect API.

Methods

The following methods apply to the global log off and disconnect API.

Method	**Description**
DisconnectSessions	Disconnects all sessions that have the designated SessionGroupId.
LogoffSessions	Logs off all sessions that have the designated SessionGroupId.
SetSessionGroupId	Sets the session's Group ID.
GetSessionHandle	Gets the handle for the current session.
SwitchSession	Sets the current session handle that selects a new session.

Method	Description
GetSessionCount	Gets the number of active started sessions.
GetSessionHandleByIndex	Gets the launched session handle using an index.
GetSessionGroupCount	Gets the number of sessions still running that belong to the specified group.

Events

The following events apply to the global log off and disconnect API.

Event	Description
OnSessionSwitch	Notifies the script that the session selected changed.
OnSessionEventPending	Notifies the script that there is at least one outstanding event for the nonselected session.
OnLogoffSessions	LogoffSessions completed successfully.
OnDisconnectSessions	DisconnectSessions completed successfully.
OnLogoffSessionsFailed	LogoffSessions failed to complete.
OnDisconnectSessionsFailed	DisconnectSessions failed to complete.

Properties

The following properties apply to the global log off and disconnect API.

Property	Description
SessionExitTimeout	Length of time to wait in seconds before judging whether or not the last disconnect or log off command failed. The default value is 60 seconds.
SessionGroupId	String that designates to which group the session belongs. Useful when the sessions need to be grouped for global log off or disconnect. This property is useful only before the connection is made. After the connection is made, use SetSessionGroupId. This property is write-only for security reasons. The default value is ICOGroup.

Error Codes

The following error codes apply to the global log off and disconnect API.

Error Name	Description
ICO_ERROR_SESSION_EXIT_TIMEOUT	Sessions failed to exit before time-out duration.
ICO_ERROR_DISCONNECT_SESSION_FAILED	DisconnectSessions API failed.
ICO_ERROR_INVALID_SESSION_HANDLE	Session handle is invalid.
ICO_ERROR_LOGOFF_SESSION_FAILED	LogoffSessions API failed.
ICO_ERROR_NO_SESSIONS_IN_GROUP	No sessions had a matching SessionGroupId.
ICO_ERROR_SESSION_ALREADY_SELECTED	Session is already selected.
ICO_ERROR_CANNOT_SCRIPT_IN_ZONE	The ICA Client Object cannot be scripted in the Internet or Restricted sites zones.

You can use ICA Client Object Global Log off and Disconnect to group launched sessions and log off or disconnect session groups. This feature is designed primarily to enable removal of all sessions simultaneously.

Sample Code

For an example script that uses the global disconnect or log off methods, see “Global Logoff and Disconnect” on page 282.

Maximum Allowed Buffer Length

For security reasons, all ICA Client Object properties and methods that are string type have a maximum allowed buffer length. See “ICA Client Object Properties” on page 204 and “ICA Browser–Specific Properties” on page 215 for the string-type properties and their maximum lengths.

Name Enumeration

Name enumeration enables you to select a server or application from the enumeration list. You can use name enumeration in a script to access names of servers, applications, and farms. To get an enumeration list, the name enumeration function must be passed a valid browser address (TCPBrowserAddress, HTTPBrowserAddress, and so on).

Methods

The following methods apply to the name enumeration operations.

Name	Description
EnumerateServers	Enumerates a list of available servers.
EnumerateApplications	Enumerates a list of available applications.
EnumerateFarms	Enumerates a list of available server farms.
GetEnumNameCount	Gets the number of enumerated names.
GetEnumNameByIndex	Gets the enumerated name by index number.
CloseEnumHandle	Closes the enumeration handle and frees memory.

For descriptions of the above methods, see “ICA Client Object APIs” on page 49. For code examples incorporating name enumeration functionality, see “Name Enumeration” on page 261.

When to Use Name Enumeration

Use the name enumeration functions to get an updated list of published resources, such as applications, servers, and server farms. Use name enumeration to employ scripting to dynamically refresh enumeration lists for published resources and notify you of updates.

Session Caching

Session caching solves problems experienced with running an ICA session within a Web browser. If the Web page containing the ICA session is modified or refreshed, the session is destroyed. Even if you try to return to the original page, you find the session no longer exists and must be reestablished.

Session caching enables users to refresh or change the Web page containing the ICA session without disrupting the ICA session. Even if you navigate to another page and then return to the original page, the session remains intact and is displayed automatically. The session caching feature introduces several new terms as listed in the table

Term	Definition
Attached	Session instance is connected and associated with an instance of the ICA Client Object.
Detached	Session instance is not associated with an instance of the ICA Client Object. The session instance is said to be *cached*.
Explicit Mode	Attaches and detaches occur when directed through scripting API.
Implicit Mode	Attaches and detaches occur automatically when starting and stopping an instance of the ICA Client Object.
Session	A single instance that can contain multiple applications.
SessionId	A unique identifier that is generated by the container and associated to each engine instance. The SessionId uniquely identifies an engine instance on the client device.
Unattached	Another term for *detached*. This implies that the session instance is not associated with an instance of the ICA Client Object. The session instance is said to be *cached*.

Features

Session caching features significantly improve the user experience of the client in a Web environment. The following sections describe the session caching features:

Time Out

A time-out value exists for cached sessions. Sessions are cached only for the duration of the time-out value. When the time-out duration is reached, the client session is disconnected from the server.

The time-out value is specified in both a global (per-user) and per-instance location. The per-instance value takes precedence over the global value. Different cached sessions can have different initial time-out values. Note that this time-out duration is the upper limit of how long the session is cached. Based on other settings or actions, the session might be logged off before the time-out. Also note that if the session is cached and then attached, the time-out value is reset to the specified value when the session is cached again. By default, the time-out is five minutes with a maximum of 120 minutes. This maximum prevents you from setting an unrealistic value.

If the time-out value is set to zero, the session is not cached. The time-out value is specified in seconds. If the global time-out value changes, it does not affect the existing cached sessions. The global change affects only new cached sessions, and only if the new cached session has no per-instance time-out specified.

Cache Limit

Only a certain number of sessions are cached. There is a limit to how many sessions can be cached per-user session. This value is specified globally for the user and cannot be overridden by a per-instance setting. This means that the user can have only a certain number of cached sessions at one time.

Though sessions are detached, they still use system resources. The cache limit is enforced to prevent detached sessions from using too much of the system resources. When a session is about to be detached, the session that has the lowest time-out value is the most likely to be deleted from the cache.

Cache limit has a default value of five cached sessions. If the value is changed while sessions are running, the changed value takes effect the next time a session needs to be cached. If the value of cache limit is set to a number smaller than the current number of cached sessions, the remove algorithm removes the difference plus one. For example, there are five sessions and the cache limit is reduced to three. Three of the five sessions must be closed to make room for the new cached session. If the limit reaches zero, it is not possible to cache any sessions.

Cache Enable

Cache enable allows you to enable and disable caching. You can enable or disable session caching globally or per-instance.The per-instance setting takes precedence.

This allows you to globally enable or disable caching, as well as to set per-session behavior. Setting the global value to False does not prevent you from setting the per-instance value to True. The global setting is used only if a per-instance setting is missing. If there is no global setting, the default is False (no caching). The Cache Enable flag can be applied only to Implicit mode. Essentially it turns automatic (implicit) features of session caching on or off. Explicit mode works regardless of the Cache Enable flag.

SessionId

SessionId is a unique identifier for the session being cached. Each session needs a unique identifier. The SessionId is an identifier that matches an instance of the ICA Client Object to a session. Without a SessionId, it is difficult to determine which session belongs to which object when attempting to reattach. There is no default value for SessionId, so a value must be specified to use session caching. If the value is not set, the session is not cached. A SessionId string must be unique; it must not be a null character. It must be greater than zero characters and fewer than 255 characters in length.

SessionId must be unique to guarantee correct placement of sessions to the respective ICA Client Objects. You can use nonunique session identifiers if you are not particular about placing a session within the right ICA Client Object container. For example, if you have two detached sessions running the same application and it does not matter which of these sessions is used, there is no need to use a truly unique identifier.

SessionId can be set but not retrieved. This prevents attaching to sessions that do not belong to the container. The container must know the SessionId to attach to the cached session.

Implicit Mode

Use Implicit mode to automatically cache and reattach sessions. It is based on settings that are configured before the object is fully loaded. This means that the parameters are specified as part of the HTML tag with the ICA Client Object. With the proper settings, the sessions are detached automatically when the page is changed or refreshed and then automatically reattached when the user returns to the original page or when the page is refreshed. There is no need to script the ICA Client Object if you are using Implicit mode. To use the ICA Client Object in Implicit mode, enable session caching and specify a valid SessionId. You must also set a nonzero time out and a suitable cache limit value.

Explicit Mode

Use Explicit mode to explicitly cache and reattach sessions. Explicit mode requires more direct control of sessions. It is possible to attach or detach sessions at any time, using scripting to control the attach and detach operations.

Application programming interfaces to do the following are available:

- Attach
- Detach
- Set SessionId
- Query if session is cached
- Get a count of cached sessions

Note Explicit mode is not affected by the Cache Enable flag.

Mixed Mode

Use Mixed mode to use the Explicit and Implicit modes together. This means that the two modes can be made to work together within an established rule set.

Implicit mode uses explicit calls internally. It is appropriate for explicit calls to be made by the script between ICA Client Object startup and shutdown functions. For example, if a **detach** is performed before ICA Client Object shutdown, the Implicit mode is informed about the detach operation and does not attempt to detach again. If Implicit mode attaches a session during ICA Client Object startup, an explicit call is allowed any time afterwards to attach a different cached session. The **attach** API automatically detaches the existing session.

ICA Client Object Startup

ICA Client Object startup provides support for automatic reattaching of cached sessions for Implicit mode.

When you create an instance of the ICA Client Object, its properties (settings) define its behavior. If caching is enabled and the SessionId is specified and valid, a search is made for a cached session corresponding to the SessionId. If a match is found, the ICA Client Object reattaches the detached session. If the attach works, the ICA Client Object ignores the setting for the Start property. This prevents problems such as starting a duplicate session. The ICA Client Object reestablishes ICA Client Object state information when the session is reattached.

ICA Client Object Shutdown

ICA Client Object shutdown provides support for automatic detaching of sessions for Implicit mode.

When an instance of the ICA Client Object is in the process of being detached, a check is made to determine if Session Caching is enabled and if a SessionId is specified and valid. If the conditions are met, the session is cached and the ICA Client Object is detached. If the limit of cached sessions is exceeded, an algorithm determines the cached session to detach. The ICA Client Object preserves state information of the detached session for use if and when the session is reattached.

Security

Security provides SessionId protection to prevent unauthorized access of the ICA Client Object.

Session caching does not allow enumeration of session identifiers, preventing unauthorized access of detached (cached) sessions by the containers. The SessionId is stored internally by, and only known to, the ICA Client Object container that created it. This protects against malicious programs or scripts that might try to reestablish sessions that are already logged on to a server. Certain environments might require that session caching be fully disabled regardless of script settings. To turn session caching on or off on a per-user basis, use SessionCacheDisable.

Methods, Properties, and Client Settings

Use the following methods, properties, and client settings to perform session caching.

Methods

These methods apply to session caching.

Name	Description
AttachSession	Attaches to a cached ICA session.
DetachSession	Detaches from an ICA session.
GetCachedSessionCount	Gets the count of cached sessions.
IsSessionAttached	Determines if the session is currently attached.
IsSessionDetached	Determines if the session is currently cached.
IsSessionRunning	Determines if the session is currently running.
SetSessionId	Sets the session ID for the current session.

Properties

These properties apply to session caching.

Name	Description
SessionCacheEnable	Enables or disables the session caching feature.
SessionCacheTimeout	Length of time in seconds to keep the cached session.
SessionId	Unique identifier for the ICA session.

Client Settings (Global Per-User)

Global settings for session caching are created in the WFCLIENT section of the user's appsrv.ini file.

Name	Description
DisableSessionCache	Enables or disables the session caching feature. This setting cannot be overridden (master switch to turn session caching on or off).
SessionCacheEnable	Enables or disables the session caching feature (default setting for client sessions).
SessionCacheTimeout	Length of time in seconds to keep the cached session.
SessionCacheLimit	Limit of how many sessions can be cached. This setting cannot be overridden.

Detailed descriptions of session caching methods and properties are included in later chapters of this book. For code examples incorporating session caching functionality, see "Session Caching" on page 256.

When to Use Session Caching

Session caching provides enhanced usability of the client in a Web environment. You can use this functionality to provide a seamless and transparent user experience. With session caching, you are ensuring that the client conforms to behavior expected from a standard Web application.

The user has the freedom to navigate between Web pages while still maintaining the connection to the server. This means that the session remains intact even when the hosting Web page is detached.

You now have the ability to allow the user to switch among multiple ICA sessions within a single ICA Client Object control window. Session caching functionality allows you to start several ICA sessions that can be detached once started. You can use scripting to select the sessions that must remain attached. The action is similar to switching channels on your television, and enables users to switch rapidly among applications within a single ICA Client Object control window.

It is also possible to create a session that has the ability to switch between several Web pages. Because the client is in a different process, it can be associated with different parents. This means it is possible to switch between several instances of Internet Explorer.

You can use session caching to have the application available and visible all the time, no matter what page you are on. The application would have a fixed position and would automatically detach and attach as you navigate between pages.

Virtual Channel Support

Virtual channel support enables you to support new devices and communication methods between the client and the server. Client-server communications take place through virtual channels, which transmit specific types of data.

Data Types

Virtual channel data can be composed of several data types. Typically, during client-server communications, data consisted only of strings transmitted in both directions. When it became possible for the server to send binary data to the client if a terminating NULL was encountered in the binary data, nothing but the NULL was received. This problem was resolved by the creation of a *binary string*. A binary string consists of binary data converted into a hexadecimal string. For example, the binary value of 0 is translated to a string such as **00**. Each binary value corresponds to a 2-byte hexadecimal string representation. Normal strings do not need to be processed in this manner, but binary data must be parsed and converted back to binary by the script.

An exception to this is the *binary* data type. In Microsoft environments, such as Visual Basic, C++, and COM, it is possible to support a binary interface using Basic String (BSTR) support. BSTRs are not null terminated and are specified in terms of their lengths. This means that BSTRs can contain binary data that does not need to be converted. BSTRs are not supported in the JavaScript environment, so the binary string data type is needed.

Data Type	**Identifier**	**Value**
String	ICO_CHANNEL_DATA_TYPE_STRING	0
Binary String	ICO_CHANNEL_DATA_TYPE_BINARY_STRING	1
Binary	ICO_CHANNEL_DATA_TYPE_BINARY	2

Methods, Properties, and Events

Use the following methods, properties and events for virtual channel support.

Methods

These methods apply to virtual channels.

Name	**Description**
CreateChannels	Creates virtual channels based on the list passed in.
GetChannelCount	Gets the number of channels created from CreateChannels.
GetChannelDataSize	Gets the size of the current incoming data.

Name	Description
GetChannelDataType	Gets the type of data for the incoming data (string or binary).
GetChannelData	Gets the data for the incoming request.
GetChannelFlags	Gets the flags for the virtual channel.
GetChannelName	Gets the name for the virtual channel by index (maximum from GetChannelCount minus one).
GetChannelNumber	Gets the virtual channel number for the virtual channel name.
GetGlobalChannelCount	Gets the total number of virtual channels for the client.
GetGlobalChannelName	Gets the name of the virtual channel by the index.
GetGlobalChannelNumber	Gets the virtual channel number of the virtual channel by name.
GetMaxChannelCount	Gets the maximum number of virtual channels supported.
GetMaxChannelRead	Gets the maximum number of bytes that can be read at once.
GetMaxChannelWrite	Gets the maximum number of bytes that can be written at once.
SendChannelData	Sets data over the virtual channel.
SetChannelFlags	Sets the flags for the virtual channel.

Properties

This property applies to virtual channels.

Name	Description
VirtualChannels	Lists of virtual channel names to create.

Events

This event applies to virtual channels.

Name	Description
OnChannelDataReceived	Notification that virtual channel data is available.

Usage

Virtual channels are used primarily for client-server communications. Most virtual channels in the client are dedicated to supporting a particular type of resource, such as printers, disks, audio, or video (desktop).

Virtual channel support in the ICA Client Object enables you to use virtual channels to create new resource types or to exchange information between client and server. An example of this is a custom application on the client communicating with a custom application on the server. Using ICA Client Object virtual channel support, you can allow exchange of any type of data without being concerned about the actual connection.

Virtual channel support in the ICA Client Object is flexible. When a client-server connection is established, almost any type of information can be exchanged. Potentially, you can support almost any resource type.

Best Practices for Creating Virtual Channels

Keep the following guidelines in mind when you create scripts that use virtual channel functions:

- You can create a maximum of 32 virtual channels. Seventeen of these are reserved for special purposes.
- Virtual channel names must not be more than seven characters in length. The first three characters are reserved for the vendor name, and the next four for the channel type. For example, **CTXAUD** represents the Citrix audio virtual channel.
- Channel names must be specified as a comma separated list. For example, if you create two channels called **OURCH1** and **OURCH2**, the string is **OURCH1,OURCH2** for CreateChannels. Using a space instead of a comma to separate channel names is also acceptable.
- Sending and receiving data enables transmission of binary streams. Though it is not easy to process binary streams with scripting from a Web page, Citrix created functionality to allow sending and receiving binary data for other containers and controls.
- If SendChannelData is specified with a data length of zero, it is assumed that the channel data is a string and the zero termination determines the length.

Detailed descriptions of the methods for virtual channel support are included in later chapters of this book. For code examples incorporating virtual channel functionality, see "Virtual Channel Support" on page 264.

ICA Client Object APIs

This chapter describes the property, methods, and events interfaces available in the ICA Client Object specification. This chapter includes the following topics:

- "Property Interface"
- "Methods Interface"
- "Events Interface"

The following table contains data types used in the descriptions. These are intended as a guide only, because the technology-specific implementations of these data types might differ.

Data Type or Value	**Remarks**
BOOLEAN	Boolean value, only TRUE or FALSE
STRING	Array of characters based on the native implementation of strings
NUMBER	Unsigned integer
VOID	No value – either no returned value or no value passed as a parameter
HANDLE	Handle to a resource

Property Interface

The property interface maintains the property list for the ICA Client Object, including adding, modifying, removing, and enumerating the contents of the property list. When the ICA connection starts, this list is used as a basis of overrides for the properties of the ICA connection to the server.

Properties are stored in key/value pairs. Both the key and value are preserved in STRING format.

Important In the following sections, the values applicable for Call Time are run-time and load-time. Load-time is the time that elapses before a connection is established to the server; run-time is when a connection is already established between the object and the server.

ClearProps

Removes all properties from the property list within the ICA Client Object.

Calling Convention

```
VOID ClearProps(VOID)
```

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Note ClearProps does not remove read-only properties.

DeleteProp

Removes Name and its corresponding value from the ICA Client Object property list. If Name does not exist, no action is taken.

Calling Convention

```
VOID DeleteProp(STRING Name)
```

Parameters

STRING Name – the name of the property to delete

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Note DeleteProp does not delete read-only properties.

DeletePropByIndex

Removes the property stored at location Index in the property list.

Note When removing a property, the index of other properties might change depending on their relative position.

Calling Convention

```
VOID DeletePropByIndex (NUMBER Index)
```

Parameters

NUMBER Index – the property index

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Note DeletePropByIndex does not remove read-only properties.

GetPropCount

Returns the number of parameters maintained by the ICA Client Object at the time the method is called.

Calling Convention

```
NUMBER GetPropCount(VOID)
```

Return value

NUMBER PropCount – the number of parameters maintained by the ICA Client Object

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

GetPropNameByIndex

Returns the name of the property stored at position Index in the property list.

Important Do not store the index of a particular property for future use. The index of a property might change relative to other properties being added or removed.

Calling Convention

```
STRING GetPropNameByIndex (NUMBER Index)
```

Return value

STRING PropName – the property name

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

GetPropValue

Returns the value for Name.

Calling Convention

```
STRING GetPropValue(STRING Name)
```

Return value

STRING PropValue – property value if successful; otherwise, zero length string

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

GetPropValueByIndex

Returns the value of the property stored at position Index in the property list.

Note Do not store the index of a particular property for future use. The index of a property might change relative to other properties being added or removed.

Calling Convention

```
STRING GetPropValueByIndex(NUMBER Index)
```

Return value

STRING PropValue – property value if successful; otherwise, zero length string

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

ResetProps

Resets the property list to its initial state. The initial state is defined by the properties started when the object was created.

Calling Convention

```
VOID ResetProps(VOID)
```

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Note ResetProps does not reset read-only properties. Current values for read-only properties are kept. The initial state of the properties does not include settings from .ica or .ini files.

SetProp

Sets a property. The operation adds the Name and Value to the property list or replaces the existing value. If the Value is NULL, any existing value is deleted. If the value is a zero length string, an existing value is replaced with the empty string or a new property is added with a zero length string value.

Calling Convention

```
VOID SetProp (STRING Name, STRING Value)
```

Parameters

STRING Name – the property name

STRING Value – the property value

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Note SetProp does not set read-only properties.

Methods Interface

The Methods Interface implements functions that control the ICA session and/or its interaction with the container.

AboutBox

Displays an About message box on the client device that contains copyright and version information about the ICA Client Object.

Calling Convention

```
VOID AboutBox(VOID)
```

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

AttachSession

Attaches the cached (detached) session to the ICA Client Object using the SessionId to match.

Calling Convention

```
NUMBER AttachSession (STRING SessionId)
```

Parameters

STRING SessionId – Unique identifier for detached session to attach

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time

Supported In

ICA Client Object 2.2 or later

Remarks

AttachSession can be called only when a session is not already attached. The SessionId should be unique to guarantee that the correct session is selected.

CloseEnumHandle

Closes the enumeration list and frees the associated memory.

Calling Convention

```
NUMBER CloseEnumHandle (HANDLE hEnum)
```

Parameters

HANDLE hEnum - handle from Enum function to close

Returns

NUMBER - ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Closes open handles when they are no longer required. Closing handles frees memory for other processes.

Connect

Connects to a server. The ICA Client Object launches an ICA session using the current connection properties. This function supports GetLastClientError.

Calling Convention

```
VOID Connect(VOID)
```

Call Time

Load-time

Supported In

ICA Client Object 2.0 or later

Note This function is asynchronous and returns to the script before the session is connected. The OnConnect or OnConnectFailed events return launch status information.

CreateChannels

Specifies the virtual channel names to be created for the session. You can specify only one parameter, but multiple virtual channel names can be included in that one parameter.

Calling Convention

```
NUMBER CreateChannels (STRING ChannelNames)
```

Parameters

STRING ChannelNames - names of virtual channels separated by a comma or spaces.

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time

Supported In

ICA Client Object 2.2 or later

Remarks

Virtual channel names specified must be a maximum of seven characters with no spaces in the name.

CreateChannels must be called only once, before connection to the server is established. Virtual channels are created during connection. To check if a particular channel was created, use GetChannelNumber.

DeleteWindow

Deletes the ICA Client Object window.

Calling Convention

```
NUMBER DeleteWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use DeleteWindow with NewWindow to create a new window for the ICA Client Object.

DetachSession

Detaches (caches) the current client session from the ICA Client Object session.

Calling Convention

```
NUMBER DetachSession (STRING SessionId)
```

Parameters

STRING SessionId – unique SessionId for cached session.

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time

Supported In

ICA Client Object 2.2 or later

Remarks

Ensure you specify a unique SessionId. If a SessionId is not specified as a parameter, DetachSession detaches the session that was previously set using SetSessionId.

DisableKeyboardInput

Disables keyboard input for the remote session.

Calling Convention

```
NUMBER DisableKeyboardInput(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

DisableKeyboardInput works only when the client is currently connected to a server. The default is to have the keyboard input enabled when the session starts. EnableMouseInput and DisableMouseInput are useful in portal environments where the need to avoid accidental input is desirable. It is also useful for monitoring sessions without worrying about input from the keyboard and mouse.

DisableMouseInput

Disables the mouse input for the session.

Calling Convention

```
NUMBER DisableMouseInput(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

DisableMouseInput works only when the client is currently connected to a server. The mouse input is enabled by default when the session starts. EnableMouseInput and DisableMouseInput are useful in portal environments where the need to avoid accidental input is highly desirable. It is also useful for monitoring sessions without worrying about input from the keyboard and mouse.

Important This function works only within the ICA Client Object window. The parent or container window (Internet Explorer or Firefox) is not affected.

DisableSizingBorder

Disables the border that allows the ICA Client Object window to be resized.

Calling Convention

```
NUMBER DisableSizingBorder(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

DisableSizingBorder is valid only when the ICA Client Object window is undocked. Use DisableSizingBorder to enforce a fixed size for the window when you want to allow the user the option to change it.

Disconnect

Disconnects the current ICA session. The user is not logged off from the server. The server determines whether to terminate or disconnect the ICA session. This function supports GetLastError and GetLastClientError.

Calling Convention

```
VOID Disconnect(VOID)
```

Call Time

Run-time

Supported In

ICA Client Object 2.0 or later

Note This function is asynchronous and returns to the script before the connection is disconnected. The OnDisconnect or OnDisconnectFailed events notify the container of disconnection status.

DisconnectSessions

Disconnects a group of sessions based on the specified session group ID.

Calling Convention

```
HANDLE DisconnectSessions(STRING SessionGroupId)
```

Parameters

STRING SessionGroupId – String that specifies the group of sessions to disconnect.

Returns

HANDLE – handle to command instance

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.3 or later

Remarks

DisconnectSessions is asynchronous. The SessionGroupId is specified with each ICA Client Object instance so that when DisconnectSessions is called, it is possible to disconnect all the sessions with the matching SessionGroupId. The handle returned by DisconnectSessions helps to determine when the command is completed. The events OnDisconnectSesssions and OnDisconnectSessionsFailed notify status.

DisplayWindow

Displays the ICA Client Object window.

Calling Convention

`NUMBER DisplayWindow(NUMBER *WndType*)`

Parameters

NUMBER WndType – Zero-based index that specifies the window type.

0	ICA Client Object window
1	Control window
2	Client window

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

DisplayWindow shows windows that were hidden using HideWindow.

DockWindow

Docks the ICA Client Object window in the control window. The control window resides on the Web page and is the logical parent to the ICA Client Object window.

Calling Convention

```
NUMBER DockWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

DockWindow places an undocked window inside a control window on the Web page.

EnableKeyboardInput

Enables keyboard input for the current session.

Calling Convention

```
NUMBER EnableKeyboardInput(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

EnableKeyboardInput works only when the client is currently connected to a server. Keyboard input is enabled by default when the session is started. EnableKeyboardInput and DisableKeyboardInput are useful in portal environments where the need to avoid accidental input is desirable. This API is also useful for monitoring sessions without being concerned about keyboard input.

EnableMouseInput

Enables mouse input for the session.

Calling Convention

```
NUMBER EnableMouseInput(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

EnableMouseInput works only when the client is currently connected to a server. Mouse input is enabled by default when the session is started. EnableMouseInput and DisableMouseInput methods are useful in portal environments where the need to avoid accidental input is desirable. This API is also useful for monitoring sessions without being concerned about mouse input.

EnableSizingBorder

Enables the border that allows the ICA Client Object window to be resized.

Calling Convention

```
NUMBER EnableSizingBorder(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

EnableSizingBorder is valid only when the ICA Client Object window is undocked. Use EnableSizingBorder to allow users the option to change the session window size.

EnumerateApplications

Lists all the published applications available for access.

Calling Convention

```
HANDLE EnumerateApplications(VOID)
```

Parameters

None

Returns

HANDLE – handle to enumeration list of applications

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

To get lists of applications for different server farms, use different browser addresses (TCPBrowserAddress or HTTPBrowserAddress).

EnumerateFarms

Lists all the server farms available for access.

Calling Convention

```
HANDLE EnumerateFarms(VOID)
```

Parameters

None

Returns

HANDLE – handle to enumerate list of farms

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

With TCP/IP, each name returned can represent a server within that farm. The farm name is separated from the server name with an asterisk (*); for example, *farmname*servername*.

EnumerateServers

Lists all the servers available for access.

Calling Convention

```
HANDLE EnumerateServers(VOID)
```

Parameters

None

Returns

HANDLE – handle to enumerate a list of servers.

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

This is the primary means of determining what the other servers within the ICA browser name space are. Ensure that you specify the address of the ICA browser using TCPBrowserAddress or HTTPBrowserAddress.

FocusWindow

Sets focus on the ICA Client Object window.

Calling Convention

```
NUMBER FocusWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

FocusWindow sets the keyboard focus to the ICA Client Object window. This ensures that all keyboard input is directed to the ICA Client Object window.

FullScreenWindow

Set the current window size to full screen.

Calling Convention

```
NUMBER FullScreenWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

FullScreenWindow forces the session to be resized to full screen. It takes a currently docked or undocked session and stretches it to fill the entire screen. It is most effective when used with the AutoAppResize property, but it can be used when DesiredHRes and DesiredVRes match the size of the screen.

GetCachedSessionCount

Returns a count of currently cached sessions.

Calling Convention

```
NUMBER GetCachedSessionCount(VOID)
```

Parameters

None

Returns

NUMBER – the number of cached sessions that exist

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetCachedSessionCount enables you to determine if an existing cached session will be deleted to make room for a new one.

GetChannelData

Returns the next data block. DataType specifies the format of the data to return.

Calling Convention

```
STRING GetChannelData(STRING ChannelName,
                      NUMBER DataType)
```

Parameters

STRING ChannelName – name of the virtual channel

NUMBER DataType – Type of data to request

0	String (string only format)
1	Binary String (string in hex format)
2	Binary (BSTR format)

Binary is not possible with Firefox. Binary is possible only when using the ICA Client Object inside another program (container). Internet Explorer does not support binary natively either.

Returns

STRING – The data is returned depending on the requested data type. If no data is available, it is returned as an empty string.

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Call GetChannelData during channel data event notification.

GetChannelDataSize

Returns the size of the next data block.

Calling Convention

```
NUMBER GetChannelDataSize(STRING ChannelName)
```

Parameters

STRING ChannelName – Name of the virtual channel

Returns

NUMBER – size of the next virtual channel block

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetChannelDataSize determines the size of the next block of data for the next read request.

GetChannelDataType

Returns the data type of the virtual channel being used for the next data block.

Calling Convention

```
NUMBER GetChannelDataType(STRING ChannelName)
```

Parameters

STRING ChannelName - name of a virtual channel

Returns

0	String (a pure string)
1	Binary (a binary string or binary)

Note In most cases, because binary is not supported in Internet Explorer, **1** represents a binary string.

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetChannelDataType provides some warning about the format of data to be received. It can also help determine the type of data to request for GetChannelData.

GetChannelCount

Returns a count of the virtual channels that were created for the current session.

Calling Convention

```
NUMBER GetChannelCount(VOID)
```

Parameters

None

Returns

NUMBER - Count of channels that were added as a result of a CreateChannels request

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetChannelCount determines the number of virtual channels that were actually created versus how many were requested.

GetChannelFlags

Returns the virtual channel flag set for a virtual channel

Calling Convention

```
NUMBER GetChannelFlags(STRING ChannelName)
```

Parameters

STRING ChannelName - name of the virtual channel for which to get flags

Returns

NUMBER - Flags for the virtual channel.

0	Do not fragment sent packets
1	Fragment sent packets

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetChannelFlags is designed to allow the script to determine what kind of features a virtual channel has. Currently, GetChannelFlags can determine only whether or not the write requests fragment automatically when too big.

GetChannelName

Returns the name of a virtual channel. Given the index of the virtual channel, the name of the virtual channel is returned. To get the highest index, use GetChannelCount and subtract one.

Calling Convention

```
STRING GetChannelName(NUMBER Index)
```

Parameters

NUMBER Index - Zero-based index used to select the virtual channel. The index can range from zero to the number of channels minus one.

Returns

STRING - the name of the virtual channel selected. An empty string indicates that the virtual channel name could not be retrieved.

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetChannelName determines which virtual channels were created during Startup.

GetChannelNumber

Returns the channel number of the specified virtual channel.

Calling Convention

```
NUMBER GetChannelNumber(STRING ChannelName)
```

Parameters

STRING ChannelName - name of the virtual channel to select

Returns

NUMBER - the actual channel number assigned to the virtual channel. This can range from 0 to 31. Because zero is reserved for system use, zero is not returned unless there is an error.

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetChannelNumber determines which channel number is assigned to a virtual channel. This is also useful in determining how many channels remain unused.

GetClientAddress

Returns the TCP/IP address of the client device based on Index.

Calling Convention

```
STRING GetClientAddress(NUMBER Index)
```

Parameters

NUMBER Index - A computer can have multiple addresses. Use GetClientAddressCount to determine how many addresses are available and use it as an upper limit minus one. The index is zero based.

Returns

STRING - Network address in STRING format

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetClientAddress permits selection of one of the TCP/IP addresses available to the client. If there is only one address, specify zero as the index. GetClientAddress enables you to acquire the client's IP address.

GetClientAddressCount

Returns the client's count of TCP/IP addresses.

Calling Convention

```
NUMBER GetClientAddressCount(VOID)
```

Parameters

None

Returns

NUMBER - Number of client addresses

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Most computers have only one TCP/IP address but some have more than one. GetClientAddressCount determines the number of TCP/IP addresses available for a specific client.

GetErrorMessage

Returns the error message text for the specified ICA Client Object error code.

Calling Convention

```
STRING GetErrorMessage(NUMBER ErrorCode)
```

Parameters

NUMBER ErrorCode - Valid ICA Client Object error code

Returns

STRING – ICA Client Object error message text

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

For a list of valid error codes that can be returned for the ICA Client Object and the Presentation Server Client, see “Error Codes” on page 297. The error message text returned is in the language supported by the Presentation Server Clients for Windows.

GetClientErrorMessage

Returns the error message text for the Presentation Server Client error code.

Calling Convention

```
STRING GetClientErrorMessage(NUMBER ErrorCode)
```

Parameters

NUMBER ErrorCode - Valid client error code

Returns

STRING - Client error message text

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

For a list of valid return codes for the ICA Client Object and the ICA Client, see “ICA Client Object Error Codes” on page 298. The error message text returned is in the language supported by the Presentation Server Clients for Windows.

GetInterfaceVersion

Returns the ICA Client Object API version as a string value. The API version has major and minor values separated by a period (for example, 2.0). GetInterfaceVersion determines what methods, properties, and events are supported by a specific version of ICA Client Object.

Calling Convention

```
STRING GetInterfaceVersion(VOID)
```

Return value

STRING Version

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

GetClientIdentification

Returns a string that completely identifies the Presentation Server Client for Windows. The format is product-model-major.minor.build-variant. For Citrix clients, the variant name is automatically set to **Citrix**. Except for the variant, all other portions of the identification string are represented by decimal numbers.

Calling Convention

```
STRING GetClientIdentification(VOID)
```

Return value

STRING ClientID

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

GetClientNetworkName

Returns the network name of the client device.

Calling Convention

```
STRING GetClientNetworkName(VOID)
```

Parameters

None

Returns

STRING - Computer network name

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

The value returned by GetClientNetworkName corresponds to the assigned network name from the operating system. For example, if the network name is **foo** for normal networking with Windows, GetClientNetworkName returns **foo**. GetClientNetworkName helps uniquely identify the network name of the client device, so it can be used with the ClientName property.

GetEnumNameCount

Returns the number of names in the enumeration list.

Calling Convention

```
NUMBER GetEnumNameCount(HANDLE hEnum)
```

Parameters

HANDLE hEnum - handle from Enum function for applications, servers, or farms

Returns

NUMBER - ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetEnumNameCount determines how many times to get information using an index.

Note Always close open handles when they are no longer required. Closing open handles frees memory for other processes.

GetEnumNameByIndex

Returns the name of an item from either the application, server, or farm enumeration list.

Calling Convention

```
STRING GetEnumNameByIndex(HANDLE hEnum,
                          NUMBER Index)
```

Parameters

HANDLE hEnum - handle from Enum function for an application, server, or farm enumeration request.

Returns

STRING - item from the enumeration list

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use GetEnumNameByIndex to get the data from the enumeration request.

Note Always close open handles when they are no longer required. Closing open handles frees memory for other processes.

GetGlobalChannelCount

Returns the total number of virtual channels in the client.

Calling Convention

```
NUMBER GetGlobalChannelCount(VOID)
```

Parameters

None

Returns

NUMBER - number of virtual channels in operation in the client

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

The current version of the Presentation Server Client can have a maximum of 32 channels. Use GetGlobalChannelCount to determine how many virtual channels are being used. GetGlobalChannelCount is not entirely helpful because there are several reserved channels that can be used only for specific virtual channel names.

GetGlobalChannelName

Returns the name of a global virtual channel.

Calling Convention

```
STRING GetGlobalChannelName(NUMBER Index)
```

Parameters

NUMBER Index - Zero-based index to specify a virtual channel. The index is based on the number of virtual channels described in GetGlobalChannelCount. Subtract one from this number because the index is zero based.

Returns

STRING - ChannelName in STRING format

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use GetGlobalChannelName to determine which channels are currently in use. It is useful in determining which channels were enabled or disabled during load time and also to determine what is running. These names include the names you create using CreateChannels.

GetGlobalChannelNumber

Returns the channel number of a global virtual channel.

Calling Convention

```
NUMBER GetGlobalChannelNumber(STRING ChannelName)
```

Parameters

STRING ChannelName - virtual channel name for which the actual virtual channel number is requested

Returns

NUMBER ChannelNumber

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetGlobalChannelNumber returns the actual virtual channel in use for the specified virtual channel name.

GetLastClientError

Returns the last error code from the Presentation Server Client.

Calling Convention

```
NUMBER GetLastClientError(VOID)
```

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.1 or later

Remarks

Returns only valid values for certain methods. All other calls generate a zero error code. For a list of ICA Client Object and client error codes, see “ICA Client Object Error Codes” on page 298.

GetLastError

Returns the last error code from the ICA Client Object.

Calling Convention

```
NUMBER GetLastError(VOID)
```

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.1 or later

Remarks

Error codes returned are valid from any ICA Client Object method. For a list of ICA Client Object and client error codes, see "ICA Client Object Error Codes" on page 298.

GetMaxChannelCount

Returns the maximum number of virtual channels.

Calling Convention

```
NUMBER GetMaxChannelCount(VOID)
```

Parameters

None

Returns

NUMBER - returns the maximum possible number of virtual channels

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetMaxChannelCount determines the maximum number of virtual channels there are on the client. Because certain virtual channels are reserved and cannot be used, use the returned count as a guideline. Currently the maximum number of virtual channels is limited to 32.

GetMaxChannelWrite

Returns the maximum number of bytes for a write operation.

Calling Convention

```
NUMBER GetMaxChannelWrite(VOID)
```

Parameters

None

Returns

NUMBER - maximum number of bytes that can be written to the server at one time

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use GetMaxChannelWrite to get an approximate estimate of the maximum allowed write request size. If the fragment flag is turned on for the virtual channel, this size is less important.

GetMaxChannelRead

Returns the maximum number of bytes for a read operation.

Calling Convention

```
NUMBER GetMaxChannelRead(VOID)
```

Parameters

None

Returns

NUMBER - maximum number of bytes that can be received from the server at one time

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetMaxChannelRead provides an estimate of how many bytes can be received at a time. This is based primarily on local buffer sizes that can be changed in the configuration but are usually left intact.

GetNotificationReason

This function returns a number representing the reason for the last event notification received by the ICA Client Object.

Calling Convention

```
NUMBER GetNotificationReason(VOID)
```

Return value

NUMBER	Reason
0	No reason available
1	Connection was successful
2	Connection failed
3	Logon succeeded
4	(Reserved)
5	Connection closed (Disconnected or Logoff)
6	Last RunPublishedApplication( ) succeeded
7	Last RunPublishedApplication( ) failed
8	ICA file present
9	ICA file download failed
10	Connection Initialization
11	Connecting to server
12	Initial properties are set
13	Disconnect operation failed
14	Log off failed
15	OnChannelDataReceived
16	OnWindowSized
17	OnWindowMoved
18	OnWindowCreated
19	OnWindowDestroyed
20	OnWindowDocked
21	OnWindowUndocked
22	OnWindowMinimized
23	OnWindowMaximized

24	OnWindowRestored
25	OnWindowFullscreened
26	OnWindowHidden
27	OnWindowDisplayed
28	OnWindowCloseRequest
29	OnDisconnectSessions
30	OnDisconnectSessionsFailed
31	OnLogoffSessions
32	OnLogoffSessionsFailed
33	OnSessionSwitch
34	OnSessionEventPending
35	OnSessionAttach
36	OnSessionDetach

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Remarks

GetNotificationReason has limited use in ICA Client Object 2.2 or later because events are called directly. However, GetNotificationReason can be used to determine what the last event was.

GetScreenColorDepth

Returns the color depth for the screen in bits per pixel for color.

Calling Convention

```
NUMBER GetScreenColorDepth(VOID)
```

Parameters

None

Returns

NUMBER - returns the color depth in bits per pixel

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetScreenColorDepth returns the color depth for the screen in bits per pixel for color. For example, 8 would mean 8 bits per pixel, which is 256 colors.

Use GetScreenColorDepth to determine the ideal match for the local desktop given the remote desktop color depth.Get the local color depth and then translate into DesiredColor.

GetSessionCount

Determines the number of active sessions within an ICA Client Object instance.

Calling Convention

```
NUMBER GetSessionCount(VOID)
```

Parameters

None

Returns

NUMBER SessionCount - sessions in an ICA Client Object instance.

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.3 or later

Remarks

GetSessionCount returns 1 if an embedded session exists and is active. If launched sessions are used, you can get many more. If GetSessionCount returns a zero, there are no active sessions.

GetSessionCounter

Returns an instantaneous count of the number of bytes or frames sent to or from the server for the current ICA session and the latency information for each counter. The value returned depends on the session counter Index selected. This function supports GetLastClientError.

Calling Convention

```
NUMBER GetSessionCounter(NUMBER Index)
```

Parameters

NUMBER	Index
0	Incoming Bytes
1	Outgoing Bytes
2	Incoming Frames
3	Outgoing Frames
4	Errors Incoming
5	Errors Outgoing
6	Last latency in milliseconds
7	Average latency in milliseconds
8	Latency deviation in milliseconds

Return value

NUMBER Counter

NUMBER Latency Information

Call Time

Run-time

Supported In

Counters 0 to 5 are supported in ICA Client Object 2.0 and 2.1, and counters 6 to 8 are supported in 2.2 or later.

Remarks

Latency is the round-trip time for a packet of information and is related to delays introduced between the client and the server. Use GetSessionCounter to determine performance of the current client-server link, and to alter your actions based on the result.

GetSessionGroupCount

Returns the number of sessions in a specific session group.

Calling Convention

```
NUMBER GetSessionGroupCount(STRING SessionGroupId)
```

Parameters

STRING SessionGroupId

Returns

NUMBER SessionGroupCount - number of sessions in a specific group.

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.3 or later

Remarks

GetSessionGroupCount determines if the logoff or disconnect methods completed. GetSessionGroupCount returns the number of sessions that have the specified group ID. The value returned by GetSessionGroupCount includes a count of all sessions the user has running in that group, including other ICA Client Object instances with the same session group name. The value of GetSessionGroupCount can be greater than the value of GetSessionCount because GetSessionCount applies only to the current ICA Client Object instance.

GetSessionHandle

Returns the currently active session handle.

Calling Convention

```
HANDLE GetSessionHandle(VOID)
```

Parameters

None

Returns

HANDLE Session - handle to the current session.

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.3 or later

Remarks

Zero is reserved and indicates that there is no current session. GetSessionHandle is useful for determining the context of certain events.

GetSessionHandleByIndex

Returns the handle of the session using an index.

Calling Convention

```
HANDLE GetSessionHandleByIndex(NUMBER Index)
```

Parameters

NUMBER Index

Returns

HANDLE Session

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.3 or later

Remarks

GetSessionHandleByIndex enumerates the different handles for sessions in an ICA Client Object instance. It is important to determine the upper limit using GetSessionCount. The index begins at zero.

GetScreenHeight

Returns the height of the screen in pixels.

Calling Convention

```
NUMBER GetScreenHeight(VOID)
```

Parameters

None

Returns

NUMBER – the height of the screen in pixels

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetScreenHeight determines the DesiredHRes property. Determine the real local desktop size and then determine the DesiredHRes of the remote desktop.

GetSessionString

Returns the session string from the server for the current ICA session. The string returned depends on the value of session string Index. This function supports GetLastClientError.

Calling Convention

```
STRING GetSessionString(NUMBER Index)
```

Parameters

NUMBER	Index
0	Server name
1	User name
2	Domain name

Call Time

Run-time

Supported In

ICA Client Object 2.0 or later

GetScreenWidth

Returns the width of the screen in pixels.

Calling Convention

```
NUMBER GetScreenWidth(VOID)
```

Parameters

None

Returns

NUMBER – the width of the screen in pixels

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetScreenWidth determines the DesiredHRes property. Determine the real local desktop size and then work out the DesiredHRes of the remote desktop.

GetSessionColorDepth

Returns the color depth for the current session in bits per pixel for color.

Calling Convention

```
NUMBER GetSessionColorDepth(VOID)
```

Parameters

None

Returns

NUMBER – Current session color depth in bits per pixel

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetSessionColorDepth determines what color depth was used to connect to the server. Make sure a session is established prior to using GetSessionColorDepth.

GetSessionHeight

Returns the height of the current session in pixels.

Calling Convention

```
NUMBER GetSessionHeight(VOID)
```

Parameters

None

Returns

NUMBER – the session height in pixels

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetSessionHeight determines the real session height. Make sure a session is established before using GetSessionHeight.

GetSessionWidth

Returns the width of the current session in pixels.

Calling Convention

```
NUMBER GetSessionWidth(VOID)
```

Parameters

None

Returns

NUMBER – the session width in pixels

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetSessionWidth determines the real session width. Make sure a session is established before using GetSessionWidth.

GetWindowHeight

Returns the height of the selected window type, in pixels.

Calling Convention

```
NUMBER GetWindowHeight(NUMBER WndType,
                                NUMBER Flags)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
1	Control Window
2	Client Window
3	Container Window

NUMBER	WndFlags
0	Inside height
1	Whole window height

Returns

NUMBER – height in pixels

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use GetWindowHeight for window management operations. You can determine sizes and placement of all the relevant windows.

The window flags determine whether the client area (inside window) or the entire window area (including border) is used.

GetWindowWidth

Returns the width of the selected window type in pixels.

Calling Convention

```
NUMBER GetWindowWIdth(NUMBER WndType,
                      NUMBER WndFlags)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
1	Control Window
2	Client Window
3	Container Window

NUMBER	WndFlags
0	Inside height
1	Whole window height

Returns

NUMBER – Returns the width of the window in pixels

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

The client window exists only if a connection to the server exists.

The window flags determine whether the client area (inside window) or the entire window area (including border) is used.

GetWindowXPosition

Returns the X position of the selected window relative to the parent window.

Calling Convention

```
NUMBER GetWindowXPosition(NUMBER WndType,
                          NUMBER WndFlags)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
	Control Window
2	Client Window
3	Container Window

NUMBER	WndFlags
0	Client area, parent relative
1	Window area, parent relative
2	Client area, screen relative
3	Window area, screen relative

Returns

NUMBER - horizontal position in pixels based on the selected flags

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

GetWindowXPosition determines the position of the relevant window types. The client window does not exist until a connection is started.

GetWindowYPosition

Returns the Y position of the selected window relative to the parent window.

Calling Convention

```
NUMBER GetWindowYPosition(NUMBER WndType,
                           NUMBER WndFlags)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
1	Control Window
2	Client Window
3	Container Window

NUMBER	WndFlags
0	Client area, parent relative
1	Window area, parent relative
2	Client area, screen relative
3	Window area, screen relative

Returns

NUMBER - the vertical position in pixels based on the requested flags

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use this API for window management operations. Use GetWindowYPosition to determine the position of specified windows.

HideWindow

Hides the specified window.

Calling Convention

```
NUMBER HideWindow(NUMBER WndType)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
1	Control Window
2	Client Window

Returns

NUMBER - ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

This method is useful for hiding the logon process until the session is fully ready. Use DisplayWindow to make the window reappear.

HideTitleBar

Hides the title bar of the ICA Client Object window.

Calling Convention

```
NUMBER HideTitleBar(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

HideTitleBar hides the title bar only when the ICA Client Object window is undocked. Also, if the title bar is hidden and the sizing border is disabled, the border disappears completely. This is a limitation of the operating system.

IsConnected

Returns the client connection state.

Calling Convention

```
BOOLEAN IsConnected(VOID)
```

Return value

TRUE if the client is connected to a server, FALSE otherwise

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

IsKeyboardInputEnabled

Determines if keyboard input is currently enabled.

Calling Convention

```
BOOLEAN IsKeyboardInputEnabled(VOID)
```

Parameters

None

Returns

BOOLEAN – TRUE(keyboard enabled), FALSE(keyboard disabled)

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

IsKeyboardInputEnabled works only when the client is currently connected to a server. Keyboard input is enabled by default when the session starts.

IsMouseInputEnabled

Determines if mouse input is currently enabled.

Calling Convention

```
BOOLEAN IsMouseInputEnabled(VOID)
```

Parameters

None

Returns

BOOLEAN – TRUE(keyboard enabled), FALSE(keyboard disabled)

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

IsMouseInputEnabled works only when the client is currently connected to a server. Mouse input is enabled by default when the session starts.

IsPassThrough

Determines if the container application is run within the server session or locally.

Calling Convention

```
BOOLEAN IsPassThrough()
```

Parameters

None

Returns

BOOLEAN – TRUE (container running on server), FALSE (container running locally)

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.4 or later

IsSessionAttached

Determines if the specified session is attached.

Calling Convention

```
BOOLEAN IsSessionAttached(STRING SessionId)
```

Parameters

STRING SessionId – Unique identifier string specifying the session

Returns

TRUE if the ICA Client Object is currently attached to the session; FALSE, if the session is missing or detached

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

IsSessionAttached checks if the session to which you are trying to attach is already attached. You can also use IsSessionAttached to check the state of sessions.

IsSessionDetached

Determines if the specified session is detached.

Calling Convention

```
BOOLEAN IsSessionDetached(STRING SessionId)
```

Parameters

STRING SessionId – Unique identifier string for the session to be queried

Returns

TRUE if the session is currently detached (cached); FALSE, if the session is missing or attached to an instance of ICA Client Object.

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

IsSessionDetached determines if the session is currently detached or not. Useful to see if a session is currently detached (cached) and available for reattachment.

IsSessionRunning

Determines if the specified session is running.

Calling Convention

```
BOOLEAN IsSessionRunning(STRING SessionId)
```

Parameters

STRING SessionId – Unique identifier string for session to be queried

Returns

TRUE if the session is running; FALSE if it is not

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

IsSessionRunning determines if the session still exists regardless of its attached or detached state.

IsWindowDocked

Determines if the ICA Client Object window is currently docked.

Calling Convention

```
BOOLEAN IsWindowDocked(VOID)
```

Parameters

None

Returns

TRUE if the ICA Client Object window is docked; FALSE if it is undocked

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

IsWindowDocked determines whether or not the window is currently docked.

LoadIcaFile

Loads an ICA file.

Calling Convention

```
VOID LoadIcaFile(STRING File)
```

Parameters

STRING File – The name of the file to load. This can be a URL or file name.

Call Time

Load-time

Supported In

ICA Client Object 2.0 or later

Remarks

LoadIcaFile initiates the asynchronous download of the ICA file from the specified location. The OnICAFile or OnICAFileFailed events notify the container about ICA file download status.

Note This function is asynchronous and returns to the script before the ICA file is loaded. The container is notified of download status by the OnICAFile or OnICAFileFailed events.

Logoff

Initiates a logoff from the server. The user's applications are terminated by the server's logoff operation and the server disconnects. This function supports GetLastClientError.

Calling Convention

```
VOID Logoff(VOID)
```

Call Time

Run-time

Supported In

ICA Client Object 2.0 or later

Remarks

The OnDisconnect or OnLogoffFailed events return logoff status notification.

Note This function is asynchronous and returns to the script before logoff occurs on the server. If logoff fails, the OnLogoffFailed event notifies the container.

LogoffSessions

Logoff a group of sessions using a session group ID.

Calling Convention

```
STRING LogoffSessions(STRING SessionGroupId)
```

Parameters

STRING SessionGroupId

Returns

HANDLE Command

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.3 or later

Remarks

LogoffSessions is similar to DisconnectSessions except that instead of disconnecting, sessions are forced to logoff. All sessions in the specified group must have the same SessionGroupId to qualify for logoff using LogoffSessions. The OnLogoffSessions and OnLogoffSessionsFailed events are used to notify of completion or error. LogoffSessions is asynchronous and should be tracked with the returned handle. SessionExitTimeout determines the duration required for logoff to complete.

MaximizeWindow

Maximizes the undocked window.

Calling Convention

```
NUMBER MaximizeWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

You can maximize the ICA Client Object window even when it is undocked. If the window is currently docked, it is undocked and maximized automatically.

MinimizeWindow

Minimizes the undocked window.

Calling Convention

```
NUMBER MinimizeWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

If the window is undocked, you can minimize it like any other main window.

NewWindow

Creates a new window for the ICA Client Object.

Calling Convention

```
NUMBER NewWindow(NUMBER XPos,
                 NUMBER YPos,
                 NUMBER Width,
                 NUMBER Height,
                 NUMBER Flags)
```

Parameters

NUMBER XPos – X position

NUMBER YPos – Y position

NUMBER Width – window width

NUMBER Height – window height

NUMBER Flags

0	Docked
1	Undocked

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

The ICA Client Object is restricted to a single window. If a window already exists, NewWindow fails.

PlaceWindowOnBottom

Places the ICA Client Object window behind other windows.

Calling Convention

```
NUMBER PlaceWindowOnBottom(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use PlaceWindowOnBottom in Web environments when the session overlaps other Web objects. This function does not provide windowless operation, which makes it less useful.

PlaceWindowOnTop

Places the ICA Client Object window in front of other windows from the same parent.

Calling Convention

```
NUMBER PlaceWindowOnTop(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

PlaceWindowOnTop ensures better visibility of the ICA Client Object window by placing it in front of other windows.

RestoreWindow

Restores the undocked window to its original size and position.

Calling Convention

```
NUMBER RestoreWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

If an undocked window is currently minimized, use RestoreWindow to restore the window to its previous state.

RunPublishedApplication

Launches a published application within a connected session. This function supports GetLastClientError.

Calling Convention

```
VOID RunPublishedApplication(STRING Name,
                             STRING Argument)
```

Parameters

STRING Name – the name of the published application

STRING Argument – the arguments for a published application

Note Argument lengths are supported differently depending on the version of Presentation Server. MetaFrame 1.8, Feature Release 1, supports up to 256 bytes of argument information, MetaFrame XP and later support up to 16,000 bytes of argument information.

A published application on the server can access the argument only if it is published with a percent sign and an asterisk (**%***) as part of its command-line specification.

Call Time

Run-time

Supported In

ICA Client Object 2.0 or later

Note This function is asynchronous and returns to the script before the application is launched. The OnPublishedApp and OnPublishedAppFailed events return the status of the launch.

ScaleDialog

Displays the scaling dialog box of the client session window. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleDialog(VOID)
```

All scale functions return a number. This number is the error code for the ICA Client Object. Potentially more information could be obtained using GetLastClientError.

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

You can use the standard scaling window on the client to specify the percentage value to scale the session by, or the exact width and height of the session window. The scaling dialog box is the same as the one used for the standard Presentation Server Clients for Windows without ICA Client Object functionality.

ScaleDisable

Disables scaling of the client session window. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleDisable(VOID)
```

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

Session scaling functionality is disabled if it is currently enabled. When scaling is disabled, you can pan and scroll the session. You can switch scaling functionality on or off as often as required – it does not affect the server in any way.

ScaleDown

Shrinks the size of the client session window. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleDown(VOID)
```

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

The current session size is reduced by ten percent, unless it is already at ten percent of original size. You can continue to reduce the session size until ten percent of the original size is reached.

ScaleEnable

Enables scaling of the client session window. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleEnable(VOID)
```

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

Session scaling functionality is enabled if it is currently disabled. Note that turning on scaling disables panning (scrolling).

ScalePercent

Scales the client session window size by a specified percentage value. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScalePercent(NUMBER Percent)
```

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

ScalePercent specifies the percentage of the scaled image in relation to the control window area. The maximum is one hundred percent and the minimum is ten percent. The percentage is based on the client window area divided by the control window area.

ScaleSize

Scales the client session window to a specified size. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleSize(NUMBER Width, NUMBER Height)
```

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

ScaleSize specifies the absolute width and height of the scaled image. The maximum size of the scaled area is the size of the control window.

ScaleToFit

Scales the client session window to fit the size of the control window. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleToFit(VOID)
```

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

ScaleToFit is the same as using ScalePercent with one hundred percent. This function directs the client to set the area of the scaled image to be the same as the control window.

ScaleUp

Enlarges the client session window. This function supports GetLastClientError.

Calling Convention

```
NUMBER ScaleUp(VOID)
```

Returns

NUMBER - ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.1 or later

Remarks

ScaleUp enlarges the current session size by ten percent, unless it is already one hundred percent.

SetSessionGroupId

Sets the SessionGroupId value for the current session.

Calling Convention

```
NUMBER SetSessionGroupId(STRING SessionGroupId)
```

Parameters

STRING SetSessionGroupId

Returns

ICORC – ICA Client Object return code

Call Time

Run-time

Supported In

ICA Client Object 2.3 or later

Remarks

SetSessionGroupId assigns a session group ID to the current session. Use SetSessionGroupId to ensure the current session has the correct group ID before calling LogoffSessions or DisconnectSessions. For security reasons, it is not possible to retrieve the session group ID.

SendChannelData

Sends data of specified length and type over the specified virtual channel.

Calling Convention

```
NUMBER SendChannelData(STRING ChannelName,
                       STRING ChannelData,
                       NUMBER DataLength,
                       NUMBER DataType)
```

Parameters

STRING ChannelName – name of the virtual channel on which to send data

STRING ChannelData – data to send over the virtual channel

NUMBER DataLength – count of bytes to send

NUMBER DataType – type of data being sent

0	String
1	Binary String
2	Binary

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use SendChannelData to send only ChannelNames created with CreateChannels. The data must match the data type. Use Binary only when hosted by an application like Visual Basic or COM.

The Binary String is encoded as a hex string representation of the original string with two characters for each byte of data. For example, 20 corresponds to decimal 32 and represents the space character.

SetChannelFlags

Sets the flags that control the specified virtual channel.

Calling Convention

```
NUMBER SetChannelFlags(STRING ChannelName,
                              NUMBER Flags)
```

Parameters

STRING ChannelName – name of the virtual channel

NUMBER Flag

0	Do not fragment packets
1	Fragment packets

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.2 or later

Remarks

SetChannelFlags enables or disables packet fragmenting. Enabling packet fragmenting allows transmission of larger packets of data.

SetSessionEndAction

Sets the action that the ICA Client Object takes when a session terminates.

Calling Convention

```
VOID SetSessionEndAction(NUMBER Action)
```

Parameters

NUMBER	Action
0	Default action does nothing at session end
1	Automatically restarts connection

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

SetSessionId

Sets a session identifier for the specified session.

Calling Convention

```
NUMBER SetSessionId(STRING SessionId)
```

Parameters

STRING SessionId – Unique identifier for the session that will use session caching

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Use SetSessionId to change the value of SessionId.

Note You cannot get the value for the SessionId after it is set.

SetWindowPosition

Sets the position of the ICA Client Object or control window relative to the parent.

Calling Convention

```
NUMBER SetWindowPosition(NUMBER WndType,
                         NUMBER XPos,
                         NUMBER YPos,
                         NUMBER WndFlags)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
1	Control Window

NUMBER XPos – Horizontal position relative to flags

NUMBER YPos – Vertical position relative to flags

NUMBER	WndFlags
0	Client area, relative to parent
1	Window area, relative to parent
2	Client area, relative to screen
3	Window area, relative to screen

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

SetWindowPosition positions the ICA Client Object window when it is undocked.

SetWindowSize

Sets the size of the ICA Client Object window or control window.

Calling Convention

```
NUMBER SetWindowSize(NUMBER WndType,
                         NUMBER Width,
                         NUMBER Height,
                         NUMBER WndFlags)
```

Parameters

NUMBER	WndType
0	ICA Client Object Window
1	Control Window

NUMBER Width – Width in pixels

NUMBER Height – Height in pixels

NUMBER	WndFlags
0	Client window size
1	Entire window size

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

SetWindowSize resizes the control or ICA Client Object windows. It is particularly useful to resize the ICA Client Object window when it is undocked.

ShowTitleBar

Displays the title bar for the ICA Client Object window.

Calling Convention

```
NUMBER ShowTitleBar(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

Usually the ICA Client Object window has a title bar. However, ShowTitleBar is useful when the title bar is explicitly hidden.

Startup

Startup is called from the container when the scripting environment is ready to receive events. Event notification is disabled until this function or any other API function is called.

Calling Convention

```
VOID Startup(VOID)
```

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.0 or later

Example

The following example illustrates how and when this function is called.

```
<html>
<head>
<title>Example of using Startup()</title>
<script language="JavaScript">
var ICO // global variable holding reference to ICA
// Client Object Connect handler
function ICO1_OnConnect()
{
alert("Connected")
}
// Page initialization

function PageInit()
{
ICO = document.ICO1// assign our object to global
ICO.Startup()
}
</script>
</head>
<body onLoad="PageInit()">
<embed type=application/x-ica
name=ICO1 width=640 height=480
address=AUS-TEST-OBJECT>
</body>
</html>
```

SwitchSession

Switches from the current session to another session based on SessionHandle.

Calling Convention

```
NUMBER SwitchSession(HANDLE hSession)
```

Parameters

HANDLE Session

Returns

NUMBER – ICA Client Object error code

Call Time

Run-time

Supported In

ICA Client Object 2.3 or later

Remarks

Switching between sessions is important for receiving events and running methods. Only the current session receives events and is manipulated by methods (except for methods that have a session handle). OnSessionSwitch notifies the script that the session context changed.

UndockWindow

Places the ICA Client Object window on the desktop.

Calling Convention

```
NUMBER UndockWindow(VOID)
```

Parameters

None

Returns

NUMBER – ICA Client Object error code

Call Time

Load-time, run-time

Supported In

ICA Client Object 2.2 or later

Remarks

UndockWindow converts a window embedded in the Web browser into one that is available on the desktop like local applications.

Events Interface

The ICA Client Object events-interface implementation is based on the underlying embedding technologies. In each of these implementations, events notification is performed through callback functions, or *handlers.*

For the ActiveX implementation, the ICA Client Object defines a set of supported events. The container defines a handler (event sink) for each of the events in which it is interested. Refer to VBScript, JavaScript, and ActiveX documentation for more information about implementing event sinks in the container.

For the Firefox plug-in implementation, the ICA Client Object defines a set of supported events. The container defines a handler for each of the events in which it is interested. The handlers are JavaScript functions, each named using the convention *ObjName_EventName*, where *ObjName* is the name of the embedded ICA Client Object, and *EventName* is the name of the event.

For example, in the following EMBED tag:

<embed type=application/x-ica width=640 height=480 name=ICO1>

the JavaScript function for the OnConnect event handler must be defined as ICO1_OnConnect.

```
function ICO1_OnConnect()

{

alert("Connected!");

}
```

The list of events supported by the ICA Client Object is as follows:

Event #	Handler
0	
1	OnConnect
2	OnConnectFailed
3	OnLogon
4	(Reserved)
5	OnDisconnect
6	OnPublishedApp
7	OnPublishedAppFailed
8	OnICAFile
9	OnICAFileFailed
10	OnInitializing

Event #	Handler
11	OnConnecting
12	OnInitialProp
13	OnDisconnectFailed
14	OnLogoffFailed
15	OnChannelDataReceived
16	OnWindowSized
17	OnWindowMoved
18	OnWindowCreated
19	OnWindowDestroyed
20	OnWindowDocked
21	OnWindowUndocked
22	OnWindowMinimized
23	OnWindowMaximized
24	OnWindowRestored
25	OnWindowFullscreened
26	OnWindowHidden
27	OnWindowDisplayed
28	OnWindowCloseRequest
29	OnDisconnectSessions
30	OnDisconnectSessionsFailed
31	OnLogoffSessions
32	OnLogoffSessionsFailed
33	OnSessionSwitch
34	OnSessionEventPending
35	OnSessionAttach
36	OnSessionDetach

Note The Reason parameter is an integer representing the event and is described by the GetNotificationReason method.

OnClick

The OnClick event callback is called when a user clicks in the embedded object's client area when an ICA session is not established. This allows the container to construct a simple user interface.

Calling Convention

```
VOID OnClick (NUMBER MouseButton,
                    NUMBER PosX,
                    NUMBER PosY,
                    NUMBER KeyMask)
```

Parameters

MouseButton generates the OnClick event. See KeyMask for value of each mouse button.

PosX and PosY are the coordinates of the mouse pointer relative to the top-left corner of the embedded object's client area. Positive X is toward the right; positive Y is toward the bottom.

KeyMask (bit-mask) can be a combination (bit-wise OR) of one or more of the values listed below. A bit is set if the corresponding mouse buttons or keys on the keyboard are pressed.

1	Left mouse button
2	Right mouse button
4	Shift key
8	Control key
16	Middle mouse button

Note An OnClick event callback is generated depending on the state of the UIActive flags. See Chapter 5, "ICA Client Object Properties"for more information.

Supported In

ICA Client Object 2.0 or later

OnConnect

The Presentation server client is successfully connected to the server. Connection occurs before the user is logged on. See OnLogon for notification of when the user is logged on successfully.

Calling Convention

```
VOID OnConnect()
```

Supported In

ICA Client Object 2.0 or later

OnConnectFailed

The connection attempt failed. Use GetLastError and GetLastClientError to determine possible causes of failure.

Calling Convention

```
VOID OnConnectFailed()
```

Supported In

ICA Client Object 2.0 or later

OnConnecting

Connecting to server notification. OnConnecting signals that initialization is complete and that the client is now attempting to connect to the server.

Calling Convention

```
VOID OnConnecting()
```

Supported In

ICA Client Object 2.0 or later

OnDisconnect

Disconnection or log off notification. When this notification event occurs, the Presentation server client is no longer available, and the ICA Client Object returns to load-time mode for the methods and properties.

Calling Convention

```
VOID OnDisconnect()
```

Supported In

ICA Client Object 2.0 or later

OnDisconnectFailed

Disconnect failure notification. A failed Disconnect method call causes this event.

Calling Convention

```
VOID OnDisconnectFailed()
```

Supported In

ICA Client Object 2.0 or later

OnICAFile

ICA file successfully downloaded notification.

Calling Convention

```
VOID OnICAFile()
```

Supported In

ICA Client Object 2.0 or later

OnICAFileFailed

ICA file download failure notification.

Calling Convention

```
VOID OnICAFileFailed()
```

Supported In

ICA Client Object 2.0 or later

OnLogon

The user is successfully logged on to the session. This event occurs during the initialization of the user's session on the server.

Calling Convention

```
VOID OnLogon()
```

Supported In

ICA Client Object 2.0 or later

OnPublishedApp

Notification that the published application launched successfully. OnPublishedApp is triggered only for RunPublishedApplication. No event is received for the initial program when it is set to a published application.

Calling Convention

```
VOID OnPublishedApp()
```

Supported In

ICA Client Object 2.0 or later

OnPublishedAppFailed

Notification of a failure to launch the published application. OnPublishedAppFailed is triggered only for RunPublishedApplication. No event is received for the initial program when it is set to a published application.

Calling Convention

```
VOID OnPublishedAppFailed()
```

Supported In

ICA Client Object 2.0 or later

OnInitializing

Connection initialization notification. OnInitializing happens right after the connection request begins. This is the stage when the client is preparing for its connection to the server.

Calling Convention

```
VOID OnInitializing()
```

Supported In

ICA Client Object 2.0 or later

OnInitialProp

Initial properties set notification. OnInitialProp signals that the initial properties are set and that these properties will be used as the basis of ResetProps.

Calling Convention

```
VOID OnInitialProp()
```

Supported In

ICA Client Object 2.0 or later

OnLogoffFailed

Notification of logoff failure. This event is a result of a failed Logoff method call.

Calling Convention

```
VOID OnLogoffFailed()
```

Supported In

ICA Client Object 2.0 or later

OnChannelDataReceived

Virtual channel data has been received.

Calling Convention

```
VOID OnChannelDataReceived(STRING ChannelName)
```

Supported In

ICA Client Object 2.2 or later

OnWindowSized

The window size has been modified.

Calling Convention

```
VOID OnWindowSized(NUMBER WndType,
                      NUMBER Width,
                      NUMBER Height)
```

Supported In

ICA Client Object 2.2 or later

OnWindowMoved

The window position was changed.

Calling Convention

```
VOID OnWindowMoved(NUMBER WndType,
                   NUMBER XPos,
                   NUMBER YPos)
```

Supported In

ICA Client Object 2.2 or later

OnWindowCreated

The window is created.

Calling Convention

```
VOID OnWindowCreated(NUMBER WndType,
                       NUMBER XPos,
                       NUMBER YPos,
                       NUMBER Width,
                       NUMBER Height)
```

Supported In

ICA Client Object 2.2 or later

OnWindowDestroyed

The window is destroyed.

Calling Convention

```
VOID OnWindowDestroyed(NUMBER WndType)
```

Supported In

ICA Client Object 2.2 or later

OnWindowDocked

The window is docked.

Calling Convention

```
VOID OnWindowDocked()
```

Supported In

ICA Client Object 2.2 or later

OnWindowUndocked

The window is undocked.

Calling Convention

```
VOID OnWindowUndocked()
```

Supported In

ICA Client Object 2.2 or later

OnWindowMinimized

The window is minimized.

Calling Convention

```
VOID OnWindowMinimized()
```

Supported In

ICA Client Object 2.2 or later

OnWindowMaximized

The window is maximized.

Calling Convention

```
VOID OnWindowMaximized()
```

Supported In

ICA Client Object 2.2 or later

OnWindowRestored

The window is restored.

Calling Convention

```
VOID OnWindowRestored()
```

Supported In

ICA Client Object 2.2 or later

OnWindowFullscreened

The window is set to full screen.

Calling Convention

```
VOID OnWindowFullscreened()
```

Supported In

ICA Client Object 2.2 or later

OnWindowHidden

The window is hidden.

Calling Convention

```
VOID OnWindowHidden(NUMBER WndType)
```

Supported In

ICA Client Object 2.2 or later

OnWindowDisplayed

The window is displayed.

Calling Convention

`VOID OnWindowDisplayed(NUMBER *WndType*)`

Supported In

ICA Client Object 2.2 or later

OnWindowCloseRequest

The window is being requested to close.

Calling Convention

```
VOID OnWindowCloseRequest()
```

Supported In

ICA Client Object 2.2 or later

Remarks

Use OnWindowCloseRequest when you want to close the undocked ICA Client Object window. This permits some control over how the ICA Client Object window is shut down. If nothing is done, the ICA Client Object window is not closed.

Alternatively, dock the window using DockWindow or delete the window using DeleteWindow.

OnDisconnectSessions

DisconnectSessions completed successfully.

Calling Convention

```
OnDisconnectSessions (NUMBER hCommand)
```

Supported In

ICA Client Object 2.3 or later

Remarks

The hCommand parameter can be matched with the original request. This way, it is possible to have multiple outstanding requests for different groups.

OnDisconnectSessionsFailed

DisconnectSessions failed.

Calling Convention

```
OnDisconnectSessionsFailed (NUMBER hCommand)
```

Supported In

ICA Client Object 2.3 or later

Remarks

The hCommand parameter helps determine which request failed. Use GetLastError and GetLastClientError to determine the nature of the problem.

OnLogoffSessions

LogoffSessions completed successfully.

Calling Convention

```
OnLogoffSessions (NUMBER hCommand)
```

Supported In

ICA Client Object 2.3 or later

Remarks

The LogoffSessions request identified by the hCommand parameter completed successfully. All sessions in that group are now logged off.

OnLogoffSessionsFailed

OnLogoffSessions failed.

Calling Convention

```
OnLogoffSessionsFailed (NUMBER hCommand)
```

Supported In

ICA Client Object 2.3 or later

Remarks

The LogoffSessions command failed. Check the error codes returned by the ICA Client Object. A common problem encountered is that time-out is reached before the sessions are fully logged off.

OnSessionSwitch

Switching between sessions.

Calling Convention

```
OnSessionSwitch (NUMBER hOldSession,
                  NUMBER hNewSession)
```

Supported In

ICA Client Object 2.3 or later

Remarks

The context of the current session changed. Either this occurred because of a call to SwitchSession or by automatic switching caused by starting and stopping of sessions. Zero is a valid handle, indicating that a current session is selected or that no current session is selected. This can occur when no sessions are active before the ICA Client Object starts or stops. It is important to pay attention to session changes.

OnSessionEventPending

Notify the script that at least one outstanding event is present for a nonselected session.

Calling Convention

```
OnSessionEventPending (NUMBER hSession,
                       NUMBER EventId)
```

Supported In

ICA Client Object 2.3 or later

Remarks

This event informs the script that there are pending events from other sessions. OnSessionEvent Pending is useful in determining when it is time to switch to another session or deciding to keep the current session active.The event ID can be mapped to the name of the event as shown in the table on page 164. Events are queued if the session is not considered active. When the session is activated, the events are released.

OnSessionAttach

The session is attached.

Calling Convention

```
OnSessionAttach (NUMBER hSession)
```

Supported In

ICA Client Object 2.3 or later

Remarks

Notifies the script that the given session is now attached. This is useful in determining when a new session is inserted into the ICA Client Object instance.

OnSessionDetach

The session is detached.

Calling Convention

```
OnSessionDetach (NUMBER hSession)
```

Supported In

ICA Client Object 2.3 or later

Remarks

Notifies the script that the given session is now detached. This is useful in determining when a session is removed from the ICA Client Object instance.

ICA Client Object Properties

This chapter describes the commonly used properties supported by the ICA Client Object and includes the following topics:

- "Accessing Properties"
- "ICA Client Object Properties"
- "ICA Browser Name Resolution"
- "About Passwords"
- "About Scaling"

Accessing Properties

Properties are accessible by using the `SetProp()` and `GetPropValue()` methods. For example:

```
obj.SetProp("DesiredHRes", "640")
obj.SetProp("Encrypt", "ON")
obj.SetProp("DesiredColor", "1")

hres=obj.GetPropValue("DesiredHres")
```

You can also set and get properties by name in the ActiveX implementation of the ICA Client Object. For example, in VBScript you can specify:

```
obj.DesiredHRes = 640
obj.Encrypt = true
obj.DesiredColor = 16Color

hres=obj.DesiredHRes
```

Note The `SetProp()` method expects property names and values in string format only. For the convenience of scripting and where supported by the particular scripting environment, well-known properties (when accessed by name) can be exposed as generic data type or enumeration; that is, in terms of integer, Boolean, and so forth. These are translated internally, as required, by the ICA Client Object.

ICA Client Object Properties

This table contains descriptions of the ICA Client Object properties and contains the types, versions in which the property first appeared, values, and size limits.

Note For security reasons, all ICA Client Object properties that are string type have a maximum allowed buffer length, which appears in this table.

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
Address	Target server address (network name or physical address).	String	2.0		2048
Application	Published application name. If defined, it overrides the Address and InitialProgram properties.	String	2.0		256
APPSRVINI	Full path and filename of appsrv.ini.	String	2.0		260
AudioBandwidthLimit	Audio bandwidth limit or audio quality.	Number	2.0	0=high, 1=medium (default) 2=low	
AUTHUsername	SSL authorization user name.	String	2.2		256
AUTHPassword	SSL authorization password.	String	2.2		256
AutoAppResize	Automatically resize the application when the window is resized.	Boolean	2.2	False (default), True	
AutoLogonAllowed	Allow auto logon to work even if the session is encrypted.	Boolean	2.2	False (default), True	
AutoScale	Automatically scale the session when the window is resized.	Boolean	2.2	False (default), True	
BackgroundColor	Specifies color of ICA Client Object background in #RRGGBB format. The “#RRGGBB” method represents a color using a triplet of hexadecimal values concatenated together. The range of each component value is 00-FF in hexadecimal (0-255 decimal.) Prefix the total value by the pound or hash (#) mark.	Number	2.0	#000000=black #FFFFFFF=white (default) other “#RRGGBB” values	
Border	Border size in pixels.	Number	2.0	0=no border 1=one pixel (default)	
BorderColor	Color of ICA Client Object border. Specified in #RRGGBB format.	Number	2.0	#000000=black (default) #FFFFFFF=white other “#RRGGBB” values	

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
BrowserProtocol	Protocol to use for ICA browser name resolution.	String	2.0	IPX, SPX, NetBIOS, UDP, HTTPonTCP	16
BrowserRetry	Retry count for browser request.	Number	2.2	3 (default)	
BrowserTimeout	Time-out period for browser request in milliseconds.	Number	2.2	1000 (default)	
CacheICAFile	Determine whether or not the ICA file will be cached by the ICA Client Object.	Boolean	2.2	TRUE (default), FALSE	
CDMAllowed	Enable support for client drive mapping.	Boolean	2.0	TRUE (default), FALSE	
ClearPassword	Clear text password for authentication.	Write-only string	2.2		256
ClientName	Unique name of client.	String	2.0	Defaults to the current ICA Client setup.	256
ClientAudio	Enable client audio support.	Boolean	2.0	FALSE (default), TRUE	
ClientPath	Path of Presentation Server Client (read-only).	Read-Only String	2.0		
ClientVersion	Version of Presentation Server Client (read-only).	Read-Only String	2.0	Format is Major.Minor.Build (example, "6.0.950")	
COMAllowed	Allow client COM port mapping.	Boolean	2.0	TRUE (default), FALSE	
Compress	Enable compression protocol.	Boolean	2.0	TRUE (default), FALSE	
Connected	Current connected state.	Read-Only Boolean	2.0	TRUE, FALSE	
ConnectionEntry	Specifies the connection entry to use in .ica or appsrv.ini file.	String	2.0		256
ControlWindowText	Text to display in control when UIActive is TRUE and window is undocked.	String	2.2		256
CPMAllowed	Allow client printer port mapping.	Boolean	2.0	TRUE (default), FALSE	

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
CustomMessage	Custom message to display in ICA Client Object when not connected. Must also set UIActive to TRUE.	String	2.0		256
Description	General description of connection.	String	2.0		256
DesiredColor	Numeric value for color depth.	Number	2.0	1=16-color (default) 2=256-color 4=16-bit-color 8=24-bit-color 16=32-bit-color	
DesiredHRes	Desired horizontal resolution of client session.	Number	2.0		
DesiredVRes	Desired vertical resolution of client session.	Number	2.0		
DisableCtrlAltDel	Do not allow a Ctrl-Alt-Del event to be sent to server.	Boolean	2.2	TRUE (default), FALSE	
Domain	Server domain used for authentication.	String	2.0		16
DoNotUseDefaultCSL	Do not use default ICA browser search technique.	Boolean	2.0	TRUE, FALSE (default)	
EnableSessionSharingClient	Enable session sharing for the client.	Boolean	2.2	FALSE (default), TRUE	
Encrypt	Enable encryption.	Boolean	2.0	TRUE (default), FALSE	
EncryptionLevelSession	Encryption level for session. Selects which encryption type to use.	String	2.0	“Encrypt”=Basic (default) “EncRC5-0”=RC5 128 bit-logon only “EncRC5-40”=RC5 40 bit “EncRC5-56”=RC5 56 bit “EncRC5-128”=RC5 128 bit	10
Height	Height of ICA Client Object (read-only).	Read-Only Number	2.0		
HTTPBrowserAddress	HTTP browser address. Used for application/server name resolution.	String	2.0		2048

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
Hotkey1Char	Hotkey 1 character (Task List).	String	2.2	"F1" (default)	16
Hotkey1Shift	Hotkey 1 Shift character (Task List).	String	2.2	"Shift" (default)	16
Hotkey2Char	Hotkey 2 character (close remote application).	String	2.2	"F3" (default)	16
Hotkey2Shift	Hotkey 2 Shift character (close remote application).	String	2.2	"Shift" (default)	16
Hotkey3Char	Hotkey 3 character (toggle title bar).	String	2.2	"F2" (default)	16
Hotkey3Shift	Hotkey 3 Shift character (toggle title bar).	String	2.2	"Shift" (default)	16
Hotkey4Char	Hotkey 4 character (Ctrl-Alt-Del).	String	2.2	"F1" (default)	16
Hotkey4Shift	Hotkey 4 Shift character (Ctrl-Alt-Del).	String	2.2	"Ctrl" (default)	16
Hotkey5Char	Hotkey 5 character (Ctrl-Esc).	String	2.2	"F2" (default)	16
Hotkey5Shift	Hotkey 5 Shift character (Ctrl-Esc).	String	2.2	"Ctrl" (default)	16
Hotkey6Char	Hotkey 6 character (Alt-Esc).	String	2.2	"F2" (default)	16
Hotkey6Shift	Hotkey 6 Shift character (Alt-Esc).	String	2.2	"Alt" (default)	16
Hotkey7Char	Hotkey 7 character (Alt-Tab).	String	2.2	"plus" (default)	16
Hotkey7Shift	Hotkey 7 Shift character (Alt-Tab).	String	2.2	"Alt" (default)	16
Hotkey8Char	Hotkey 8 character (Alt-Backtab).	String	2.2	"minus" (default)	16
Hotkey8Shift	Hotkey 8 Shift character (Alt-Backtab).	String	2.2	"Alt" (default)	16
Hotkey9Char	Hotkey 9 character (Ctrl-Shift-Esc).	String	2.2	"F3" (default)	16
Hotkey9Shift	Hotkey 9 Shift character (Ctrl-Shift-Esc).	String	2.2	"Ctrl" (default)	16
Hotkey10Char	Hotkey10 character (toggle latency reduction).	String	2.2	"F4" (default)	16

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
Hotkey10Shift	Hotkey 10 Shift character (toggle SpeedScreen Latency Reduction).	String	2.2	"Ctrl" (default)	16
ICAFile	Name of ICA file to use for connection. URLs are allowed.	String	2.0		2048
ICAPortNumber	TCP/IP port number to use for connecting to the server.	Number	2.0	1494 (default), other valid port numbers	
ICASOCKSProtocolVersion	Version of SOCKS being used.	Number	2.2	-1 (default), 0 = Auto detect, 5	
ICASOCKSProxyHost	Server address for SOCKS port.	String	2.2		2048
ICASOCKSProxyPortNumber	Port number for SOCKS support.	Number	2.2	1080 (default)	
ICASOCKSRFC1929UserName	SOCKS authorization user name.	String	2.2		256
ICASOCKSRFC1929Password	SOCKS authorization password.	String	2.2		256
ICASOCKSTimeout	Time-out for SOCKS request in milliseconds.	Number	2.2	5000 (default)	
IconIndex	Index into the IconPath file for associated icon.	Number	2.0		
IconPath	Icon file pathname.	String	2.0		260
InitialProgram	Initial program to run on server.	String	2.0		260
IPXBrowserAddress	IPX browser address for application server name resolution.	String	2.0		2048
KeyboardTimer	Queue keyboard input for a specified number of milliseconds.	Number	2.0	0 (default)	
LanaNumber	NetBIOS LANA number.	Number	2.2	0 (default), other numbers	
Launch	Launch the session instead of embedding it.	Boolean	2.0	TRUE, FALSE (default)	
LocHTTPBrowserAddress	Local HTTP browser address for application/ server name resolution.	String	2.0		2048

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
LocIPXBrowserAddress	Local IPX browser address for application/server name resolution.	String	2.0		2048
LocNetbiosBrowserAddress	Client local NetBIOS browser address for application/server name resolution.	String	2.0		2048
LocTCPBrowserAddress	Local TCP/IP browser address for application/ server name resolution.	String	2.0		2048
LogAppend	Append to existing log file.	Boolean	2.0	TRUE, FALSE (default)	
LogConnect	Log connection/ disconnection events.	Boolean	2.0	TRUE (default), FALSE	
LogErrors	Log errors from connection.	Boolean	2.0	TRUE (default), FALSE	
LogFile	Name of log file.	String	2.0		260
LogFlush	Flush out log results for each write (very slow). All the log data is written as quickly as possible instead of being cached in a memory. This ensures that the log file is completely up to date at any given moment.	Boolean	2.0	TRUE, FALSE (default)	
LogKeyboard	Log the keystrokes.	Boolean	2.0	TRUE, FALSE (default)	
LogReceive	Log the receive data.	Boolean	2.0	TRUE, FALSE (default)	
LogTransmit	Log the transmit data.	Boolean	2.0	TRUE, FALSE (default)	
MODULEINI	Pathname for module.ini file (optional).	String	2.0		260
MouseTimer	Queue mouse input for a specified number of milliseconds.	Number	2.0	0 (default)	
NetbiosBrowserAddress	NetBIOS browser address. Used for application/server name resolution.	String	2.0		2048
NotificationReason	Reason for last event notification.	Read-Only Number	2.0	See event table	

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
Password	User password for connection. Internal user based on PasswordType. All passwords are stored in encrypted form. See “About Scaling” on page 218.	Write-Only String	2.0		256
PasswordType	Type of password being used. See “About Scaling” on page 218.	Number	2.0	0=Unencrypted (default) 1=Encrypted	
PersistentCacheEnabled	Enable persistent cache (image disk caching).	Boolean	2.0	TRUE, FALSE (default)	
ProtocolSupport	List of protocol drivers to load (advanced).	String	2.0		1024
Reliable	Enable reliable protocol support.	Boolean	2.0	TRUE (default), FALSE	
ScalingHeight	Height of scaled window. See “About Scaling” on page 218.	Number	2.0	0 (default), value in pixels >=0	
ScalingMode	Mode of scaling. See “About Scaling” on page 218.	Number	2.0	0=Disabled (default), 1=Percent, 2=Size, 3=Size to fit	
ScalingPercent	Percent of scaling to control window size. See “About Scaling” on page 218.	Number	2.0	100 (default), percentage value >=0	
ScalingWidth	Width of scaled window. See “About Scaling” on page 218.	Number	2.0	0=default, value in pixels >= 0	
ScreenPercent	Specifies how large the launched session should be as a percentage of total screen size.	Number	2.2	1 to 100	
Scrollbars	This property enables or disables scroll bars on the session window. Scrollbars are enabled if set to TRUE and disabled if set to FALSE. Scrollbars are enabled by default.	Boolean	2.0	TRUE (default), FALSE	
SessionCacheEnable	Enables or disables automatic session caching.	Boolean	2.2	FALSE (default), TRUE	
SessionCacheTimeout	Duration of time in seconds before cached session times out.	Number	2.2	300 (default) < 7200	

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
SessionEndAction	Specify what to do when session ends.	Number	2.0	0=Nothing (default) 1=Restart	
SessionId	Session identifier for session caching.	Write-only string	2.2		256
SessionSharingLaunchOnly	Launch an application only within an existing session.	Boolean	2.2	FALSE (default), TRUE	
SessionSharingName	Name of the session to be shared.	String	2.2		256
SSLCACert0, SSLCACert1… SSLCACertN	CA certificates.	String	2.2		256
SSLCiphers	SSL ciphers.	String	2.2	ALL, COM, and GOV	3
SSLCommonName	SSL common name.	String	2.2		2048
SSLEnable	Enable or disable SSL support.	Boolean	2.2	FALSE (default), TRUE	
SSLNoCACerts	Number of CA certificates available.	Number	2.2		
SSLProxyHost	Address of SSL proxy server.	String	2.2	"*" (default)	2048
Start	Start the client session immediately.	Boolean	2.0	TRUE, FALSE (default)	
TCPBrowserAddress	TCP/IP browser address. Used for application/server name resolution.	String	2.0		2048
TextColor	Color of text for ICA Client Object specified in #RRGGBB format.	Number	2.0	#000000=black (default) #FFFFFFF=white other "#RRGGBB" values...	
Title	Session window title.	String	2.0	"Server" (default)	256
TransportDriver	Transport used for connection.	String	2.0	TCP/IP (default), IPX, NetBIOS	8
TransportReconnectDelay	Time to wait before attempting automatic reconnect for launched session.	Number	2.2	30 (default)	
TransportReconnectEnabled	Enable or disable automatic client reconnect for launched session.	Boolean	2.2	TRUE (default), FALSE	

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
TransportReconnectRetries	Number of times to attempt automatic reconnect for launched session.	Number	2.2	FALSE (default), TRUE	
TWIMode	Flag for selecting seamless mode for launched session.	Boolean	2.2	FALSE (default), TRUE	
UIActive	Enable custom UI.	Boolean	2.0	TRUE, FALSE (default)	
UpdatesAllowed	Enable client updates from the server.	Boolean	2.0	TRUE (default), FALSE	
UseAlternateAddress	Allows the use of an alternate address for firewall connections.	Boolean	2.2	FALSE (default), TRUE	
UserName	User name for authentication.	String	2.0		256
Version	Version of ICA Client Object.	Read-only string	2.0	Current ICA Client Object version number	
VirtualChannels	Specifies the virtual channels to be opened on connection. You can specify multiple channel names as a comma separated list. Names must be restricted to seven characters or less.	String	2.2		260
VSLAllowed	Enable client print spooling.	Boolean	2.0	TRUE (default), FALSE	
WFCLIENTINI	Full path and file name of the wfclient.ini file.	String	2.0		260
Width	Width of ICA Client Object (read-only).	Read-only number	2.0		
WinstationDriver	WinStation driver.	String	2.0	“ICA 3.0” (default)	8
WorkDirectory	Default starting directory on server.	String	2.0		260
XmlAddressResolutionType	Address resolution used for XML requests.	String	2.2	“DNS-Port” (default), “IPv4-Port”	10
EnableSessionSharingHost	Enable session sharing host mode.	Boolean	2.3	FALSE(default), TRUE	
IPCLaunch	Use Inter Process Communication (IPC) to launch session.	Boolean	2.3	TRUE(default), FALSE	

Name	Description	Type	Ver.	Valid Value(s)	Max. Buffer Length
LongCommandLine	Parameters for InitialProgram published application.	String	2.3		2048
OutputMode	Output mode for the client engine.	Number	2.3	0(non-headless), 1(normal), 2(renderless), 3(windowless)	
Session	Interface for client automation object.	Interface	2.3		
SessionExitTimeout	Time in seconds to wait for LogoffSessions or DisconnectSessions to complete before triggering failed event.	Number	2.3	60(default)	
SessionGroupId	Session group ID for the session.	String	2.3		256
TWIDisableSessionSharing	Disable session sharing for seamless sessions.	Boolean	2.3	FALSE(default)	

ICA Browser Name Resolution

The ICA browser is a component of Presentation Server that resolves server and published application names into network addresses. The Presentation Server Client requests this service during the initial connection attempt.

The Presentation Server Client uses a number of ways to communicate with the ICA browser. Usually it is through the same protocol as the stated transport driver. This can differ at times if more than one type of communication is supported on that protocol.

Communication regarding name resolution between the client and server takes place automatically if the client and server are on the same subnet. If the client is able to broadcast a resolution request over the local subnet and get a response, there is no need to specify where the ICA browser is.

If the client and the server are not on the same subnet, specify exactly how this communication will occur to guarantee proper name resolution.

Specific ICA Client Object properties that you can use to do this are listed in the following section.

ICA Browser–Specific Properties

The following table lists ICA Client Object properties that relate specifically to the ICA browser.

Note For security reasons, all ICA Client Object properties that are string type have a maximum allowed buffer length, which is displayed in this table.

Name	Description	Type	Possible Value(s)	Max. Buffer Length
BrowserProtocol	Protocol to use for ICA browser name resolution.	String	IPX, SPX, NETBIOS, UDP, HTTPonTCP	
DoNotUseDefaultCSL	Do not use default ICA browser search technique.	Boolean	TRUE, FALSE (default)	
LocHTTPBrowserAddress	Local HTTP browser address. Used for application/server name resolution.	String		2048
LocIPXBrowserAddress	Local IPX browser address. Used for application/server name resolution.	String		2048

Name	Description	Type	Possible Value(s)	Max. Buffer Length
LocNetbiosBrowserAddress	Local NetBIOS browser address. Used for application/ server name resolution.	String		2048
LocTCPBrowserAddress	Local TCP/IP browser address. Used for application/server name resolution.	String		2048

Usage

The DoNotUseDefaultCSL property controls which set of addresses to use.

When set to FALSE (the default), these addresses are used:

- HTTPBrowserAddress
- IPXBrowserAddress
- NetbiosBrowserAddress
- TCPBrowserAddress

When set to TRUE, these addresses are used:

- LocHTTPBrowserAddress
- LocIPXBrowserAddress
- LocNetbiosBrowserAddress
- LocTCPBrowserAddress

The difference between the two (addresses used when either TRUE or FALSE) is that the addresses used when DoNotUseDefaultCSL is TRUE are strong overrides and the client does not use any other internal address from the .ini files.

The addresses used when DoNotUseDefaultCSL is FALSE do not take precedence over what might be in the [WFCLIENT] section of the client .ini files.

Note **Loc** addresses are more specific and there is less chance of confusion about name resolution.

The BrowserProtocol determines which protocol to use to contact the ICA browser:

- IPX and SPX use IPXBrowserAddress or LocIPXBrowserAddress
- NetBIOS uses NetbiosBrowserAddress or LocNetbiosBrowserAddress
- UDP and HTTPOnTCP use TCP/IP but differ in the type of ICA browser communication.
    - UDP uses the traditional technique of using UDP over a TCP/IP network.
    - HTTPOnTCP uses TCP with the HTTP protocol. HTTPOnTCP selects the browser address from LocHTTPBrowserAddress or HTTPBrowserAddress. UDP selects from TCPBrowserAddress or LocTCPBrowserAddress.

ICA Browser Address Grid

TransportDriver	**BrowserProtocol**	**DoNotUseDefaultCSL**	**ICA Browser Address Used**
IPX	IPX	FALSE	IPXBrowserAddress
IPX	IPX	TRUE	LocIPXBrowserAddress
SPX	SPX	FALSE	IPXBrowserAddress
SPX	SPX	TRUE	LocIPXBrowserAddress
NetBIOS	NetBIOS	FALSE	NetbiosBrowserAddress
NetBIOS	NetBIOS	TRUE	LocNetbiosBrowserAddress
TCP/IP	UDP	FALSE	TCPBrowserAddress
TCP/IP	UDP	TRUE	LocTCPBrowserAddress
TCP/IP	HTTPonTCP	FALSE	HTTPBrowserAddress
TCP/IP	HTTPonTCP	TRUE	LocHTTPBrowserAddress

Usage

HTTPonTCP is popular to use on the Internet because it allows traffic to pass through firewalls. Without this feature, it is difficult to publish applications on the Internet because most firewalls will prohibit UDP traffic.

You can usually use the non-Loc browser addresses. The only time it becomes a serious issue is when there are previous installations of the client and the client was configured to have specific browser addresses that do not correspond to the properties set by the ICA Client Object.

About Passwords

Normally, passwords are set using plain text; however, it is possible to use a Citrix encrypted password instead. The PasswordType property allows you to determine whether a Citrix encrypted or a plain text password is being used.

To enable encryption of a plain text password, set PasswordType to 0. The ICA Client Object automatically encrypts the password so it is ready to pass to the client.

If you set the Password property to encrypted text and set PasswordType to 1, the ICA Client Object does not modify the Password property before passing it to the client.

To access encrypted Citrix passwords, generate them using Program Neighborhood or Remote Application Manager, then extract them from the corresponding .ini or .ica file. The encrypted version of the password is numerical and approximately twice as long as the original string length.

To write a log on script, use a form to include fields like username, password, and domain name. Submit these values to the ICA Client Object using SetProp. Do not set PasswordType if you are dealing only with passwords input by users. The default behavior is to use plain text and convert it to encrypted form.

The Password property is the only property that is write-only. This means that you cannot read its value from a script. If it is a requirement to keep the password, do not expect the ICA Client Object to provide it.

For security reasons, copies of plain text passwords are not retained by the ICA Client Object. Password references used during initialization are cleared from memory before it is released.

About Scaling

To determine the initial scaled state of the session, use scaling properties: ScalingMode, ScalingPercent, ScalingHeight, and ScalingWidth.

To take advantage of scaling, set DesiredHRes and DesiredVRes values to be larger than the control window width and height. For example, if the DesiredHRes/DesiredVRes is 1024x768, set the control window size to 640x480. Setting ScalingMode to 3 forces the 1024x768 session to fit in a 640x480 window.

ScalingMode is the most important of the available scaling properties. It specifies the scaling mode that is used for the initial connection. ScalingMode can be set to one of four possible initial states. These are:

State	Meaning	Description
0	Disabled	This is the default setting and means that scaling is not enabled at initialization.
1	Percent	This setting instructs the ICA Client Object to use the ScalingPercent property to determine the size of the scaling area. It ignores ScalingWidth and ScalingHeight. One hundred percent means that the area of the scaling is the same as the area of the control window. Fifty percent means that the scaling area is 50 percent of the control window.
2	Size	This setting instructs the ICA Client Object to use the ScalingHeight and ScalingWidth properties. It ignores the ScalingPercent property. The width and height of the scaling area are checked against the size of the control window. The size cannot be bigger than the control window area.
3	To fit window	This setting instructs the ICA Client Object to fit the session into the existing control window. This is the easiest to do for a script because it forces the session to show its complete yet scaled area inside the control window. This mode ignores the three other properties ScalingPercent, ScalingWidth, and ScalingHeight.

These properties are the initial settings. Changes made to properties during a connected session do not have any effect. When the session is established, use scaling methods to change the scaling attributes of the session. For information about scaling methods, see “Methods Interface” on page 59.

Example Scripts

This chapter contains examples of how the ICA Client Object works and explains how the embedding and scripting interfaces contribute to its functionality.

This chapter contains the following types of examples:

- “Embedding”
- “Scripting”
- “Logon Methods”
- “Custom User Interfaces”
- “Mouse Click”
- “Scaling Methods”
- “Client-Side HTML/VBScripting Using ActiveX”
- “Client-Side HTML/VBScripting Using the Firefox Plug-In”
- “Session Caching”
- “Name Enumeration”
- “Automatic Application Resizing”
- “Client Network Name and Address”
- “Error Message Text from Error Codes”
- “Global Logoff and Disconnect”

Embedding

The following sample HTML scripts illustrate how to embed the ICA Client Object in a Web page.

ActiveX

In this example, an ICA session is embedded in a Web page. The client session automatically connects to the remote server when the user accesses the page.

- An instance of the ICA Client Object is embedded in the Web page using the OBJECT tag. ClassId is a unique identifier for the ActiveX object, commonly known as the *GUID*. This identifier must be specified as shown in the following example.
- The codebase defines the URL where a binary copy of the ActiveX object is located if the ActiveX object is not already installed on the client device. The ActiveX embedding model provides the default mechanism to download the object if required.
- A published application (ActiveXDemo.ica) as defined by the IcaFile property is launched within this ICA session.
- Two of the properties for the embedded session are specified using the PARAM tag, border size, and automatic start, respectively.

```
<html>
<head>
<title>ICA Client Object Demo - embedded ActiveX</title>
</head>
<body>
<center>
<object
classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
codebase="http://icaobj_server/icaobj/wfica.cab#version=-1,-1,-1,-1"
width="640" height="480">
<param name="IcaFile"
value="http://icaobj_server/icaobj/ActiveXDemo.ica">
<param name="Border" value="1">
<param name="Start" value="Auto">
</object>
</center>
</body>
</html>
```

Firefox Plug-in

In this example, an ICA session is embedded in a Web page. The client automatically connects to the remote server when the page is first accessed.

- An instance of the ICA Client Object is embedded in the Web page using the `EMBED` tag. The `Type` property is a unique MIME-type identifier for the plug-in object. This identifier must be specified as illustrated below.
- A published application (PluginDemo.ica) as defined by the IcaFile property is launched within this ICA session.

```
<html>
<head>
<title>ICA Client Object Demo - embedded Plugin</title>
</head>
<body>
<center>
<embed
    width="640" height="480"
    type="application/x-ica"
    name="icaObj"
    IcaFile="http://icaobj_server/icaobj/PluginDemo.ica"
    Border="1"
    Start="Auto">
</embed>
</center>
</body>
</html>
```

Scripting

The following HTML scripts illustrate how the ICA Client Object can be controlled through scripting when embedded in a Web page.

Important You must set the **Address** property to an existing server.

ActiveX

The following parts of the example illustrate the event interface.

- An alert box appears when the client connects to the server (OnConnect event)
- An alert box appears when the user is successfully logged on to the server (OnLogon events)
- An alert box appears when the user is disconnected from the server (OnDisconnect event)

```
<html>
<head>
<title>Event Example for ActiveX</title>
<script language="javascript" type="text/javascript">
<!--
function PageInit()
{
    document.icaobj.Startup();
}
-->
</script>
<script id="clientEventHandlersJS" language="javascript"
type="text/javascript">
<!--
function icaobj_OnConnect()
{
    alert("OnConnect");
}
function icaobj_OnDisconnect()
{
    alert("OnDisconnect");
}
function icaobj_OnLogon()
{
    alert("OnLogon");
}
//-->
</script>
<script language="javascript" for="icaobj" event="OnConnect"
type="text/javascript">

<!--
 icaobj_OnConnect();
//-->
</script>
<script language="javascript" for="icaobj" event="OnDisconnect"
```

<contd ...>

<contd ...>

```
<!--
 icaobj_OnConnect();
//-->
</script>
<script language="javascript" for="icaobj" event="OnDisconnect"

type="text/javascript">
<!--
 icaobj_OnDisconnect();
//-->
</script>
<script language="javascript" for="icaobj" event="OnLogon"
type="text/javascript">
<!--
 icaobj_OnLogon();
//-->
</script>
</head>
<body onload="PageInit()">
<center>
<h1>Event Example for ActiveX</h1>
<p>Event notification for Connect, Disconnect and Logon.</p>

<object
  id="icaobj" style="WIDTH: 400px; HEIGHT: 300px"
  classid="clsid:238F6F83-B8B4-11CF-8771-00A024541EE3">
  <param name="Address" value="mfserver">
</object>
</center>
</body>
</html>
```

Firefox Plug-in

The following parts of the example illustrate the event interface.

- An alert box appears when the client connects to the server (OnConnect event)
- An alert box appears when the user is successfully logged on to the server (OnLogon events)
- An alert box appears when the user is disconnected from the server (OnDisconnect event)

```
<html>
<head>
<title>Event Example for Firefox Plugin</title>
<script language="javascript" type="text/javascript">
<!--
function icaobj_OnConnect()
{
alert("OnConnect");
}
function icaobj_OnDisconnect()
{
alert("OnDisconnect");
}
function icaobj_OnLogon()
{
    alert("OnLogon");
}
function PageInit()
{
    document.icaobj.Startup();
}
-->
</script>
</head>
<body onload="PageInit()">
<center>
<h1>Event Example for Firefox Plugin</h1>
<p>Event notification for Connect, Disconnect and Logon.</p>

<embed
  name="icaobj"
  type="application/x-ica"
  width="400" height="300"
  address="mfserver">
</center>
</body>
</html>
```

Logon Methods

The following examples illustrate use of a Web page to establish a desktop session on the server. When you click the Logon button, a desktop session with the specified credentials is initiated on the server.

ActiveX

This is the ActiveX logon example.

```
<html>
<head><title>Logon Example for ActiveX
</title>
<script language="javascript" type="text/javascript">
<!--
function Login(form)
{
    document.icaobj.SetProp("Address",form.address.value);
    document.icaobj.SetProp("Username",form.username.value);
    document.icaobj.SetProp("Password",form.password.value);
    document.icaobj.SetProp("Domain",form.domain.value);
    document.icaobj.Connect();
}
-->
</script></head>
<body>
<center>
<h1>Logon Example for ActiveX</h1>
<p>Logon with Username and Password.</p>
<object
  id="icaobj" style="WIDTH: 400px; HEIGHT: 300px"
  classid="clsid:238F6F83-B8B4-11CF-8771-00A024541EE3">
</object>
<form name="loginform">
  Server: <input type="text" name="address"><br>
  Username: <input type="text" name="username"><br>
  Password: <input type="password" name="password"><br>
  Domain: <input type="text" name="domain"><br>
  <input type="button" value="Login" onclick="Login(this.form)">
</form>
</center>
</body>
</html>
```

Firefox Plug-in

This is the Firefox plug-in logon example.

```
<html>
<head>
<title>Logon Example for Firefox Plugin</title>
<script language=“javascript” type=“text/javascript”>
<!--
function Login(form)
{
    document.icaobj.SetProp(“Address”,form.address.value);
    document.icaobj.SetProp(“Username”,form.username.value);
    document.icaobj.SetProp(“Password”,form.password.value);
    document.icaobj.SetProp(“Domain”,form.domain.value);
    document.icaobj.Connect();
}
-->
</script>
</head>
<body>
<center>
<h1>Logon Example for Firefox Plugin</h1>
<p>Logon with Username and Password.</p>

<embed
  name=“icaobj”
  type=“application/x-ica”
  width=“400” height=“300”>

<form name=“loginform”>
  Server: <input type=“text” name=“address”><br>
  Username: <input type=“text” name=“username”><br>
  Password: <input type=“password” name=“password”><br>
  Domain: <input type=“text” name=“domain”><br>
<input type=”button” value=”Login” onclick=”Login(this.form)”>
</form>

</center>
</body>
</html>
```

Custom User Interfaces

The following examples illustrate how to customize the appearance of the user interface for the client session prior to connection.

Important You must set the **UIActive** property to **true** for CustomMessage to work.

ActiveX

This example changes the border size and color, background color, and message text and color of the user interface prior to connection.

```
<html>
<head>
<title>UI Example for ActiveX</title>
</head>
<body>
<center>
<h1>UI Example for ActiveX</h1>
<p>Custom border, background and message text.</p>

<object
  id="icaobj" style="WIDTH: 400px; HEIGHT: 300px"
  classid="clsid:238F6F83-B8B4-11CF-8771-00A024541EE3">
  <param name="UIActive" value="true">
  <param name="Border" value="5">
  <param name="BorderColor" value="#ff0000">
  <param name="BackgroundColor" value="#0000ff">
  <param name="TextColor" value="#00ff00">
  <param name="CustomMessage"
value="Custom UI Example: Border size and color;
Background color; Message Text Color.">
</object>

</center>
</body>
</html>
```

Firefox Plug-in

This example changes the border size and color, background color, and message text and color of the user interface prior to connection.

```
<html>
<head>
<title>Custom UI Example for Firefox Plugin</title>
</head>
<body>
<center>
<h1>Custom UI Example for Firefox Plugin</h1>
<p>Custom border, background and message text.</p>

<embed name=“icaobj”
  type=“application/x-ica”
  width=“400” height=“300”
  UIActive=“true”
  Border=“10”
  BorderColor=“#ff0000”
  BackgroundColor=“#0000ff”
  TextColor=“#00ff00”
CustomMessage=“Custom UI Example: Border size and color;
Background color; Message Text Color.”>
</center>
</body>
</html>
```

Mouse Click

The following examples illustrate handling of mouse input.

Note You must set the **UIActive** property to **true** to enable the OnClick event.

ActiveX

This ActiveX example defines how mouse input is handled.

```
<html>
<head>
<title>UI OnClick Example for ActiveX</title>
<script id="clientEventHandlersJS" language="javascript"
type="text/javascript">
<!--
function icaobj_OnClick(mbutton, x, y, keymask)
{
  var mButtonStr;
  var keyMaskStr = "";
switch (mbutton)
  {
    case 1:
      mButtonStr = "left";
      break;
    case 2:
      mButtonStr = "right";
      break;
    case 16:
      mButtonStr = "middle";
      break;
  }
  if (keymask == 0)
  {
    keyMaskStr = "none";
  }
  else
  {
    if (keymask & 4)
    {
      keyMaskStr += "Shift ";
    }
```

<contd ...>

<contd ...>

```
if (keymask & 8)
    {
      keyMaskStr += “Ctrl”;
}
}
alert(“You clicked the ” + mButtonStr + “ mouse button\nat the follow-
ing location (“ + x + ”, “ + y + ”),\nkeymask=“ + keyMaskStr + ”.”);
}

//-->
</script>
<script language=“javascript” for=“icaobj”
event=“OnClick(mbutton, x, y, keymask)” type=“text/javascript”>
<!--
 icaobj_OnClick(mbutton, x, y, keymask)

//-->
</script>
</head>
<body>
<center>
<h1>UI OnClick Example for ActiveX</h1>

<p>UIActive, custom message text, and OnClick handling.</p>

<object
  id=“icaobj” style=“WIDTH: 400px; HEIGHT: 300px”
  classid=“clsid:238F6F83-B8B4-11CF-8771-00A024541EE3”>
  <param name=“UIActive” value=“true”>
  <param name=“CustomMessage” value=“Hold down the Shift
and/or Ctrl key, and click mouse buttons anywhere...”>
</object>

</center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example defines how mouse input is handled.

```
<html>
<head>
<title>UI OnClick Example for Firefox Plugin</title>
<script language="javascript" type="text/javascript">
<!--
function icaobj_OnClick(mbutton, x, y, keymask)
{
  var mButtonStr;
  var keyMaskStr = "";

  switch (mbutton)
  {
    case 1:
      mButtonStr = "left";
      break;
    case 2:
      mButtonStr = "right";
      break;
    case 16:
      mButtonStr = "middle";
      break;
  }
  if (keymask == 0)
  {
    keyMaskStr = "none";
  }
  else
  {
    if (keymask & 4)
    {
      keyMaskStr += "Shift ";
    }
    if (keymask & 8)
    {
      keyMaskStr += "Ctrl";
    }
  }
alert("You clicked the " + mButtonStr + " mouse button\nat
the following location (" + x + ", " + y +"),
\nkeymask=" + keyMaskStr + ".");
}
```

<contd ...>

<contd...>

```
//
// Page initialization function
//
function PageInit()
{
    document.icaobj.Startup();
}

//-->
</script>
</head>
<body onload="PageInit()">
<center>
<h1>UI OnClick Example for Firefox Plugin</h1>
<p>UIActive, custom message text, and OnClick handling.</p>

<embed name="icaobj"
  type="application/x-ica"
  width="400" height="300"
  UIActive="true" CustomMessage="Hold down the Shift and/or Ctrl key,
and click mouse buttons anywhere...">
</center>
</body>
</html>
```

Scaling Methods

The following sections contain usage examples of each scaling method using ActiveX and Firefox plug-in.

ScaleEnable and ScaleDisable

ScaleEnable and ScaleDisable enable or disable scaling of the client session window. When scaling is disabled, you can pan and scroll the session. Enabling scanning disables panning.

ActiveX

This ActiveX example illustrates the ScaleEnable and ScaleDisable scaling methods.

```
<html>
<head>
   <title>ScaleEnable/ScaleDisable</title>
</head>
<body>
<h1>ScaleEnable/ScaleDisable</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
id=ICO1 width="640" height="480">
  <param name="Address" value="MyServer">
  <param name="DesiredHRes" value="800">
  <param name="DesiredVRes" value="600">
</object>
<br>
<form>
<input type="button" value="ScaleEnable"
onClick="document.ICO1.ScaleEnable()">
<input type="button" value="ScaleDisable"
onClick="document.ICO1.ScaleDisable()">
</form>
</center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the ScaleEnable and ScaleDisable scaling methods.

```
<html>
<head>
   <title>ScaleEnable/ScaleDisable</title>
</head>
<body>
<h1>ScaleEnable/ScaleDisable</h1>
<center>
<br>
<embed  name=ICO1
type=application/x-ica
width=640 height=480
Address=MyServer
DesiredHRES=800 DesiredVRES=600>
<br>
<form>
<input type=“button” value=“ScaleEnable”
onClick=“document.ICO1.ScaleEnable()”>
<input type=“button” value=“ScaleDisable”
onClick=“document.ICO1.ScaleDisable()”>
</form>
</center>
</body>
</html>
```

ScaleUp and ScaleDown

ScaleUp enlarges the client session window and ScaleDown shrinks the size of the client session window.

ActiveX

This ActiveX example illustrates the ScaleUp and ScaleDown scaling methods.

```
<html>
<head>
   <title>ScaleUp/ScaleDown</title>
</head>
<body>
<h1>ScaleUp/ScaleDown</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3" id=ICO1
width="640" height="480">
  <param name="Address" value="MyServer">
  <param name="DesiredHRes" value="800">
  <param name="DesiredVRes" value="600">
  <param name="ScalingMode" value="3">
</object>
<br>
<form>
<input type="button" value="ScaleDown"
onClick="document.ICO1.ScaleDown()">
<input type="button" value="ScaleUp"
onClick="document.ICO1.ScaleUp()">
</form>
</center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the ScaleUp and ScaleDown scaling methods.

```
<html>
<head>
   <title>ScaleUp/ScaleDown</title>
</head>
<body>
<h1>ScaleUp/ScaleDown</h1>
<center>
<br>
<embed  name=ICO1
type=application/x-ica
width=640 height=480
Address=MyServer
DesiredHRES=800 DesiredVRES=600
ScalingMode=3>
<br>
<form>
<input type="button" value="ScaleDown"
onClick="document.ICO1.ScaleDown()">
<input type="button" value="ScaleUp"
onClick="document.ICO1.ScaleUp()">
</form>
</center>
</body>
</html>
```

ScalePercent

ScalePercent scales the client session window size by a specified percentage value.

ActiveX

This ActiveX example illustrates the ScalePercent scaling method.

```
<html>
<head>
   <title>ScalePercent</title>
</head>
<body>
<h1>ScalePercent</h1>
<center>
<br>
<object
classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3" id=ICO1
width="640" height="480">
      <param name="Address" value="MyServer">
      <param name="DesiredHRes" value="800">
      <param name="DesiredVRes" value="600">
      <param name="ScalingMode" value="3">
</object>
<br>
<form>
<input name=percent  type="text" size=4 value="100">
<input type=button   value="ScalePercent"
onClick="document.ICO1.ScalePercent
(parseInt(this.form.percent.value))">
<br>
</form>
</center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the ScalePercent scaling method.

```
<html>
<head>
   <title>ScalePercent</title>
</head>
<body>
<h1>ScalePercent</h1>
<center>
<br>
<embed  name=ICO1
type=application/x-ica
width=640 height=480
Address=MyServer
DesiredHRES=800 DesiredVRES=600
ScalingMode=3>
<br>
<form>
<input name=percent  type="text" size=4 value="100">
<input type=button value="ScalePercent"
onClick="document.ICO1.ScalePercent
(parseInt(this.form.percent.value))">
<br>
</form>
</center>
</body>
</html>
```

ScaleToFit

ScaleToFit scales the client session window to fit the size of the control window.

ActiveX

This ActiveX example illustrates the ScaleToFit scaling method.

```
<html>
<head>
   <title>ScaleToFit</title>
</head>
<body>
<h1>ScaleToFit</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
id=ICO1 width="640" height="480">
      <param name="Address" value="MyServer">
      <param name="DesiredHRes" value="800">
      <param name="DesiredVRes" value="600">
</object>
<br>
<form>
<input type=button value="ScaleToFit"
onClick="document.ICO1.ScaleToFit()">
</form>
</center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the ScaleToFit scaling method.

```
<html>
<head>
   <title>ScaleToFit</title>
   <script>
   </script>
</head>
<body>
<h1>ScaleToFit</h1>
<center><br>
<embed  name=ICO1
        type=application/x-ica
        width=640 height=480
        Address=MyServer
        DesiredHRES=800 DesiredVRES=600
        ScalingMode=0>
<br>
<form>
<input type=“button” value=“ScaleToFit”
onClick=“document.ICO1.ScaleToFit()”>
<br>
</form>
</center>
</body>
</html>
```

ScaleDialog

ScaleDialog displays the scaling dialog box of the client session window.

ActiveX

This ActiveX example illustrates the ScaleDialog scaling method.

```
<html>
<head>
   <title>ScaleDialog</title>
</head>
<body>
<h1>ScaleDialog</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
id=ICO1 width="640" height="480">
      <param name="Address" value="MyServer">
      <param name="DesiredHRes" value="800">
      <param name="DesiredVRes" value="600">
</object>
<br>
<form>
<input type=button value="ScaleDialog"
onClick="document.ICO1.ScaleDialog()">
</form>
</center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the ScaleDialog scaling method.

```
<html>
<head>
   <title>ScaleDialog</title>
</head>
<body>
<h1>ScaleDialog</h1>
<center><br>
<embed  name=ICO1
        type=application/x-ica
        width=640 height=480
        Address=MyServer
        DesiredHRES=800 DesiredVRES=600
        ScalingMode=3><br>
<form>
<input type=“button” value=“ScaleDialog”
onClick=“document.ICO1.ScaleDialog()”>
</form>
</center>
</body>
</html>
```

Scaling Properties

In the following examples, for ActiveX and the Firefox plug-in respectively, scaling properties of a session are set using various scaling modes and associated scaling properties.

Example 1

In the following example, ScalingMode=0. In this mode, session scaling is disabled.

ActiveX

This ActiveX example illustrates the scaling properties.

```
<html>
<head></head>
<body>
<h1>Embed with ScalingMode=0 (Disabled)</h1>
<center><br>
<object classid=“clsid:238f6f83-b8b4-11cf-8771-00a024541ee3”
      id=ICO1 width=“640” height=“480”>
      <param name=“Address” value=“MyServer”>
      <param name=“DesiredHRes” value=“800”>
      <param name=“DesiredVRes” value=“600”>
      <param name=“ScalingMode” value=“0”>
</object><br></center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the scaling properties.

```
<html>
<head></head>
<body>
<h1>Embed with DesiredHRES=800 DesiredVRES=600 width=640 height=480
ScalingMode=0 (Disabled)</h1>
<center><br>
<embed  name=ICO1
        type=application/x-ica
        width=640 height=480
        Address=MyServer
        DesiredHRES=800 DesiredVRES=600
        ScalingMode=0>
<br></center>
</body>
</html>
```

Example 2

In the following example, ScalingMode=1. In this mode, scaling size is specified in percentage values. The percentage by which the session is scaled is specified by ScalingPercent.

ActiveX

This ActiveX example illustrates the scaling properties.

```
<html>
<head></head>
<body>
<h1>Embed with ScalingMode=1 ScalingPercent=50</h1>
<center><br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
      id=ICO1 width="640" height="480">
      <param name="Address"    value="MyServer">
      <param name="DesiredHRes"    value="800">
      <param name="DesiredVRes"    value="600">
      <param name="ScalingMode"    value="1">
      <param name="ScalingPercent" value="50">
</object><br></center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the scaling properties.

```
<html>
<head></head>
<body>
<h1>Embed with DesiredHRES=800 DesiredVRES=600 width=640 height=480
ScalingMode=1 ScalingPercent=50</h1>
<center> <br>
<embed  name=ICO1
        type=application/x-ica
        width=640 height=480
        Address=MyServer
        DesiredHRES=800 DesiredVRES=600
        ScalingMode=1 ScalingPercent=50>
<br></center>
</body>
</html>
```

Example 3

In the following example, ScalingMode=2. In this mode, scaling size is specified by height and width values. The height and width by which the session is scaled is specified by ScalingWidth and ScalingHeight.

ActiveX

This ActiveX example illustrates the scaling properties.

```
<html>
<head></head>
<body>
<h1>Embed with ScalingMode=2 ScalingWidth=320 ScalingHeight=240</h1>
<center><br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
      id=ICO1 width="640" height="480">
      <param name="Address"       value="MyServer">
      <param name="DesiredHRes"   value="800">
      <param name="DesiredVRes"   value="600">
      <param name="ScalingMode"   value="2">
      <param name="ScalingWidth"  value="320">
      <param name="ScalingHeight" value="240">
</object><br></center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the scaling properties.

```
<html>
<head>
</head>
<body>
<h1>Embed with DesiredHRES=800 DesiredVRES=600 width=640 height=480
ScalingMode=2 ScalingWidth=320 ScalingHeight=240</h1>
<center><br>
<embed  name=ICO1
        type=application/x-ica
        width=640 height=480
        Address=MyServer
        DesiredHRES=800 DesiredVRES=600
        ScalingMode=2 ScalingWidth=320 ScalingHeight=240>
<br></center>
</body>
</html>
```

Example 4

In the following example, ScalingMode=3. In this mode, the session is sized to fit into the existing control window.

ActiveX

This ActiveX example illustrates the scaling properties.

```
<html>
<head></head>
<body>
<h1>Embed with ScalingMode=3 (SizeToFit)</h1>
<center><br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
      id=ICO1 width="640" height="480">
      <param name="Address"        value="MyServer">
      <param name="DesiredHRes"    value="800">
      <param name="DesiredVRes"    value="600">
      <param name="ScalingMode"    value="3">
</object>
<br></center>
</body>
</html>
```

Firefox Plug-in

This Firefox plug-in example illustrates the scaling properties.

```
<html>
<head>
</head>
<body>
<h1>Embed with DesiredHRES=800 DesiredVRES=600 width=640 height=480
ScalingMode=3 (SizeToFit)</h1>
<center><br>
<embed  name=ICO1
        type=application/x-ica
        width=640 height=480
        Address=MyServer
        DesiredHRES=800 DesiredVRES=600
        ScalingMode=3>
<br></center>
</body>
</html>
```

Client-Side HTML/VBScripting Using ActiveX

The following examples illustrates how scripting is used in client-side HTML/VBScript to allow the user to start different applications:

- Embed the ICA Client Object (icaObj) using the **OBJECT** tag.
- Use the **SELECT** list (AppList) to present a selection of available applications. In this case, actual .ica files are used to identify the published application.
- Press the **Start** button to disconnect the current ICA session using the icaObj.Disconnect method. The selected .ica file is then used as the parameter to the icaObj.SetProp method, which sets the value of the IcaFile property internal to the icaObj. The selected application is then started with the icaObj.Connect method.
- Use the **Connect** button to call the icaObj.Connect method to (re)connect to the server.
- Use the **Disconnect** button to call the icaObj.Disconnect method to disconnect from the server.
- Use the **Logoff** button to call the icaObj.Logoff method to terminate the current application and log off from the server.
- Use the **About** button to call the icaObj.AboutBox method to display information about the currently installed Presentation Server Clients for Windows.

Additionally, you can use the Events Notification feature to enhance the user experience by disabling buttons that do not apply to the current state of the ICA session. For example, the **Connect** button can be disabled when the container (browser) is notified that the connection is established, and subsequently reenabled when it is notified that the connection is closed or lost due to network problems.

```
<html>
<head>
<title>ICA Client Object Demo - scripting ActiveX</title>
<script id="clientEventHandlersVBS" language="vbscript">
<!--
Sub ConnButton_onclick
  '(re)connect current session
  icaObj.Connect
End Sub
Sub DiscButton_onclick
  'disconnect current session
  icaObj.Disconnect
End Sub
Sub LogoffButton_onclick
  'logoff from remote server
  icaObj.Logoff
End Sub
Sub AboutButton_onclick
  'show ICA Client Object's "AboutBox"
  icaObj.AboutBox
End Sub
Sub StartButton_onclick
  'Disconnect current application and start the selected application
  if icaObj.IsConnected then
    icaObj.Disconnect
  end if
  icaObj.SetProp "ICAFile",
AppList.options(AppList.selectedIndex).text
  icaObj.Connect
End Sub
-->
</script>
</head>
<body><center>
<table cellspacing="1" cellpadding="10" border="0">
  <tr><td align="center">
<object id="icaObj"
codebase="http://icaobj_server/icaobj/wfica.cab#version=-1,-1,
-1,-1" classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
width="320" height="240">
</object>
</td></tr>
```

<contd ...>

<contd ...>

```
<tr><td align=“center”>
<input id=“ConnButton” type=“button” value=“Connect”
name=“ConnButton”>
<input id=“DiscButton” type=“button” value=“Disconnect”
name=“DiscButton”>
<input id=“LogoffButton” type=“button” value=“Logoff”
name=“LogoffButton”>
<input id=“AboutButton” type=“button” value=“About” name=“AboutBut-
ton”>
</td></tr>
<tr><td>
    <hr style=“LEFT: 10px; TOP: 104px”>
    Available Applications:<br>
    <select id=“AppList” style=“WIDTH: 500px”>
    <option selected>http://icaobj_server/icaobj/excel.ica</option>
    <option>http://icaobj_server/icaobj/word.ica</option>
    <option>http://icaobj_server/icaobj/ActiveXDemo.ica</option>
    </select>
<input id=“StartButton” type=“button” value=“Start”
name=“StartButton”><br>
<input id=“LogoffButton” type=“button” value=“Logoff”
name=“LogoffButton”><br>
<input id=“AboutButton” type=“button” value=“About”
name=“AboutButton”>
</td></tr>

<tr><td>
    <hr style=“LEFT: 10px; TOP: 104px”>
    Available Applications:<br>
<select id=“AppList” style=“WIDTH: 500px”>
    <option selected>http://icaobj_server/icaobj/excel.ica</option>

<option>http://icaobj_server/icaobj/word.ica</option>
<option>http://icaobj_server/icaobj/ActiveXDemo.ica</option>
</select>
<input id=“StartButton” type=“button” value=“Start”
name=“StartButton”><br>
</td></tr>
</table>

</center>
</body>
</html>
```

Client-Side HTML/VBScripting Using the Firefox Plug-In

The following example illustrates how simple scripting is used in client side HTML/JavaScript to allow the user to start different published applications.

- Embed the ICA Client Object (icaObj) using the EMBED tag.
- Use a **SELECT** list (AppList) to present a selection of available applications. In this case, actual .ica files are used to identify published applications.
- Press the **Start** button to disconnect the current ICA session using the icaObj.Disconnect method. The selected .ica file is then used as the parameter to the icaObj.SetProp method, which sets the value of the IcaFile property internal to the icaObj. The selected application is then started by the icaObj.Connect method.
- Select the **Connect** button to call the icaObj.Connect method to (re)connect to the server.
- Use the **Disconnect** button to call the icaObj.Disconnect method to disconnect from the server.
- Use the **Logoff** button to call the icaObj.Logoff method to terminate the current application and log off from the server.
- Use the **About** button to call the icaObj.AboutBox method to display information about the currently installed Presentation Server Client for Windows.

```
<html>
<head>
<title>ICA Client Object sample - scripting Firefox Plugin</title>
<script language=“JavaScript” type=“text/javascript”>
<!--

function ConnButton_onclick(form)
{
    var icaObj = form.icaObj
    // (re)connect current session
    icaObj.Connect()
}

function DiscButton_onclick(form)
{
```

<contd ...>

```
 var icaObj = form.icaObj
    // disconnect current session
    icaObj.Disconnect()
}
function LogoffButton_onclick(form)
{
    var icaObj = form.icaObj
    // logoff from remote server
    icaObj.Logoff()
}
function AboutButton_onclick(form)
{
    var icaObj = form.icaObj
    // show ICA Client Object's “AboutBox”
    icaObj.AboutBox()
}
function StartButton_onclick(form)
{
    var icaObj = form.icaObj
    var AppList = form.AppList
    // Disconnect current application and start the selected
//application
    if (icaObj.IsConnected())
        icaObj.Disconnect()
    icaObj.SetProp(“IcaFile”,
AppList.options[AppList.selectedIndex].text)
    icaObj.Connect()
}
// -->
</script>
</head>
<body>
<center>
<form name=“appsForm”>
<table cellspacing=“1” cellpadding=“10” border=“0”>
<tr><td align=“center”>
  <embed
    name=“icaObj”
    type=“application/x-ica”
    width=“320” height=“240”>
</td></tr>
```

<contd ...>

<contd ...>

```
<tr><td align="center">
  <input id="ConnButton" type="button" value="Connect" name="ConnBut-
ton" onclick="ConnButton_onclick(this.form)">&nbsp;
  <input id="DiscButton" type="button" value="Disconnect" name="Dis-
cButton" onclick="DiscButton_onclick(this.form)">&nbsp;
  <input id="LogoffButton" type="button" value="Logoff"
name="LogoffButton" onclick="LogoffButton_onclick(this.form)">&nbsp;
  <input id="AboutButton" type="button" value="About" name="AboutBut-
ton" onclick="AboutButton_onclick(this.form)">
</td></tr>

<tr><td>
  <hr style="LEFT: 10px; TOP: 104px">
  Available Applications:<br>
  <select name="AppList" style="WIDTH: 500px">
  <option selected>http://icaobj_server/icaobj//excel.ica</option>
  <option>http://icaobj_server/icaobj/word.ica</option>
  <option>http://icaobj_server/icaobj/PluginDemo.ica</option>
  </select>
  <input id="StartButton" type="button" value="Start" name="StartBut-
ton" onclick="StartButton_onclick(this.form)"><br>
</td></tr>

</table>
</form>
</center>
</body>
</html>
```

Session Caching

This example illustrates the use of session caching methods and properties.

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
  <title>Session Cache example</title>
    <script language="JavaScript">
    <!--
function Attach(form)
    {
        form.AttachRC.value =
ICO1.AttachSession(form.AttachSessionId.value)
        form.AttachClientRC.value = ICO1.GetLastClientError()
    }
    function Detach(form)
    {
        form.DetachRC.value =
ICO1.DetachSession(form.DetachSessionId.value)
        form.DetachClientRC.value = ICO1.GetLastClientError()
}
    function IsAttached(form)
    {
        form.IsAttachedBool.value =
ICO1.IsSessionAttached(form.IsAttachedSessionId.value)
        form.IsAttachedClientRC.value = ICO1.GetLastClientError()
}
    function IsDetached(form)
    {
        form.IsDetachedBool.value =
ICO1.IsSessionDetached(form.IsDetachedSessionId.value)
        form.IsDetachedClientRC.value = ICO1.GetLastClientError()
}
    function IsRunning(form)
    {
        form.IsRunningBool.value =
ICO1.IsSessionRunning(form.IsRunningSessionId.value)
        form.IsRunningClientRC.value = ICO1.GetLastClientError()
    }
```

<contd ...>

<contd ...>

```
function SetId(form)
    {
        form.SetSessionIdRC.value =
ICO1.SetSessionId(form.SetSessionId.value)
        form.SetSessionIdClientRC.value = ICO1.GetLastClientError()
    }
    function SetAddress(form)
    {
        ICO1.SetProp("Address", form.Address.value)
    }
    function SetSessionIdProp(form)
    {
        ICO1.SetProp("SessionId", form.SessionId.value)
    }
    function SetSessionCacheEnable(form)
    {
        ICO1.SetProp("SessionCacheEnable",
form.SessionCacheEnable.value)
    }
    function MyStartup()
    {
        document.ICO1.Startup()
        SetAddress(SessionCache)
        SetSessionIdProp(SessionCache)
        SetSessionCacheEnable(SessionCache)
    }
//  -->
   </script>
</head>
<body onLoad="MyStartup()">
<h1>Session Cache example</h1>
<center>
<object
    id="ICO1"
    classid="clsid:238F6F83-B8B4-11cf-8771-00A024541EE3"
    height=480 width=640>
    <param name=SessionId value="MyId">
    <param name=SessionCacheEnable value=True>
</object>
</center>
```

<contd ...>

<contd ...>

```
<form name=SessionCache>
<center>
<table border cellpadding=3>
<tr>
    <th>Property</th>
    <th>Value</th>
    <th>Set</th>
</tr>
<tr>
    <td>Address</td>
    <td><input type=text name=Address size=20 value=192.1.1.128></td>
    <td><input type=button value=“Set Address”
onClick=SetAddress(this.form)></td>
</tr>
<tr>
    <td>SessionId</td>
    <td><input type=text name=SessionId size=20 value=MyId></td>
    <td><input type=button value=“Set SessionId”
onClick=SetSessionIdProp(this.form)></td>
</tr>
<tr>
    <td>SessionCacheEnable</td>
    <td><input type=text name=SessionCacheEnable size=5
value=TRUE></td>
    <td><input type=button value=“Set SessionCacheEnable”
onClick=SetSessionCacheEnable(this.form)></td>
</tr>
</table>
<table border cellpadding=3>
<tr>
    <th>Function</th>
    <th>Parameters</th>
    <th>Run</th>
    <th>ICO Result</th>
    <th>Client Result</th>
</tr>
<tr>
    <td>AttachSession</td>
    <td>SessionId:<input type=text name=AttachSessionId size=20></td>
    <td><input type=button value=AttachSession
onClick=“Attach(this.form)”></td>
    <td>ICORC: <input type=text name=AttachRC size=4></td>
    <td>ClientRC: <input type=text name=AttachClientRC size=4></td>
</tr>
```

<contd ...>

<contd ...>

```
<tr>
    <td>DetachSession</td>
    <td>SessionId:<input type=text name=DetachSessionId size=20></td>
    <td><input type=button value=DetachSession
onClick="Detach(this.form)"></td>
    <td>ICORC: <input type=text name=DetachRC size=4></td>
    <td>ClientRC: <input type=text name=DetachClientRC size=4></td>
</tr>
<tr>
    <td>GetCachedSessionCount</td>
    <td>&nbsp;</td>
    <td><input type=button value=GetCachedSessionCount
onClick="GetCachedCount(this.form)"></td>
    <td>Count: <input type=text name=Count size=4></td>
    <td>ClientRC: <input type=text name=CountClientRC size=4></td>
</tr>
<tr>
    <td>IsSessionAttached</td>
    <td>SessionId:<input type=text name=IsAttachedSessionId
size=20></td>
    <td><input type=button value=IsSessionAttached
onClick="IsAttached(this.form)"></td>
    <td>BOOL: <input type=text name=IsAttachedBool size=4></td>
    <td>ClientRC: <input type=text name=IsAttachedClientRC
size=4></td>
</tr>
<tr>
    <td>IsSessionDetached</td>
    <td>SessionId:<input type=text name=IsDetachedSessionId
size=20></td>
    <td><input type=button value=IsSessionDetached
onClick="IsDetached(this.form)"></td>
    <td>BOOL: <input type=text name=IsDetachedBool size=4></td>
    <td>ClientRC: <input type=text name=IsDetachedClientRC
size=4></td>
</tr>
<tr>
    <td>IsSessionRunning</td>
    <td>SessionId:<input type=text name=IsRunningSessionId
size=20></td>
<td><input type=button value=IsSessionRunning
onClick="IsRunning(this.form)"></td>
<td>BOOL: <input type=text name=IsRunningBool size=4></td>
    <td>ClientRC: <input type=text name=IsRunningClientRC size=4></td>
</tr>
```

<contd ...>

<contd ...>

```
<tr>
    <td>SetSessionId</td>
    <td>SessionId:<input type=text name=SetSessionId size=20></td>
    <td><input type=button value=SetSessionId
onClick="SetId(this.form)"></td>
    <td>ICORC: <input type=text name=SetSessionIdRC size=4></td>
    <td>ClientRC: <input type=text name=SetSessionIdClientRC
size=4></td>
</tr>
</table>
</form>
</center>
</body>
</html>
```

Name Enumeration

This example illustrates the use of methods and properties for name enumeration operations.

```
<!doctype html public “-//w3c//dtd html 4.0 transitional//en”>
<html>
<head>
  <meta http-equiv=“Content-Type” content=“text/html;
charset=iso-8859-1”>
  <title>Enumeration Example</title>
    <script language=“javascript”>
    <!--
    function EnumServers(form)
    {
        var hndEnum
        SetEnumValues(form)
        hndEnum = document.ICO1.EnumerateServers()
        Enum(hndEnum)
    }
    function EnumApps(form)
    {
        var hndEnum
        SetEnumValues(form)
        hndEnum = document.ICO1.EnumerateApplications()
        Enum(hndEnum)
    }
function EnumFarms(form)
    {
        var hndEnum
        SetEnumValues(form)
        hndEnum = document.ICO1.EnumerateFarms()
        Enum(hndEnum)
    }
function SetEnumValues(form)
    {
        document.ICO1.SetProp(“BrowserProtocol”,
form.ProtocolType.value)
        document.ICO1.SetProp(“TCPBrowserAddress”,
form.BrowserAddress.value)
        document.ICO1.SetProp(“HTTPBrowserAddress”,
form.BrowserAddress.value)
    }
```

<contd...>

<contd...>

```
function Enum(hndEnum)
    {
        var EnumCnt
        var i
        var EnumName
        if(hndEnum != 0)
        {
            EnumCnt = document.ICO1.GetEnumNameCount(hndEnum)
            if(EnumCnt == 0)
            {
                alert("No names to enumerate")
            }
            for(i = 0; i < EnumCnt; i++)
            {
                EnumName = document.ICO1.GetEnumNameByIndex(hndEnum,i)
                alert("index "+i+" Name "+EnumName)
            }
            document.ICO1.CloseEnumHandle(hndEnum)
        }
        else
        {
            alert("did not get enum handle")
        }
    }
    function MyStartup()
    {
        document.ICO1.Startup()
    }
    //-->
   </script>
</head>
<body onLoad="MyStartup()">
<h1>Enumeration Example</h1>

<center>
<br>
<object
  id="ICO1"
  classid="clsid:238F6F83-B8B4-11cf-8771-00A024541EE3"
  height=480 width=640>
    <param name="Address" value="192.1.1.128">
</object>
```

<contd...>

<contd…>

```
<br>
</center>
<form name=results>
<center>
Protocol:<input type=text value=UDP      name=ProtocolType
size=20><br>
BrowserAddress:<input type=text value="192.1.1.128"
name=BrowserAddress size=20><br>
<input type="button" value="Enumerate Servers"
onClick="EnumServers(this.form)"><br>
<input type="button" value="Enumerate Applications"
onClick="EnumApps(this.form)"><br>
<input type="button" value="Enumerate Farms"
onClick="EnumFarms(this.form)"><br>
</form>
</body>
</html>
```

Virtual Channel Support

This example illustrates the use of methods and properties available for creating virtual channels.

```
<!doctype html public “-//w3c//dtd html 4.0 transitional//en”>
<html>
<head>
  <meta http-equiv=“Content-Type” content=“text/html;
charset=iso-8859-1”>
  <title>Examples of Virtual Channel methods</title>
    <script langauge=“javascript” for=“ICO1”
event=“OnChannelDataReceived(ChannelName)”
    type=“text/javascript”>
    <!--
    ICO1_OnChannelDataReceived(ChannelName);
    //-->
    </script>
  <script language=“JavaScript”>
    <!--
    function ICO1_OnConnect()
    {
        document.results.ChannelName.value = “”
        document.results.PktCount.value = 0
        document.results.TotalPktSize.value = 0
    }
    function ICO1_OnDisconnect()
    {
        document.results.ChannelName.value = “”
        document.results.PktCount.value = 0
        document.results.TotalPktSize.value = 0
    }
function ICO1_OnChannelDataReceived(ChannelName)
    {
        var DataType
        var DataSize
        var ChannelData
        var rc
        var ClientRC

        DataType = document.ICO1.GetChannelDataType(ChannelName)
        DataSize = document.ICO1.GetChannelDataSize(ChannelName)
        if(DataSize != 0)
```

<contd...>

<contd...>

```
{
            ChannelData = document.ICO1.GetChannelData(ChannelName,
DataType)
            document.results.ReadData.value = ChannelData
            document.results.ChannelName.value = ChannelName
            document.results.BytesRecv.value = DataSize
            document.results.DataType.value = DataType
            document.results.PktCount.value =
parseInt(document.results.PktCount.value) + 1
            document.results.TotalPktSize.value =
parseInt(document.results.TotalPktSize.value) + DataSize
        }
        rc = document.ICO1.SendChannelData(ChannelName, ChannelData,
DataSize, DataType)
        ClientRC = document.ICO1.GetLastClientError()
        document.results.VCForm.value = rc
        document.results.CltErr.value = ClientRC
    }
    function WriteData(form)
    {
        form.VCForm.value =
  document.ICO1.SendChannelData(form.SendChannel.value,
  form.SendData.value,0,0)
        form.CltErr.value = document.ICO1.GetLastClientError()
    }
    function GetChannelInfo(form)
    {
        var cnt
        var i
        var ChannelName
        var ChannelNum
        cnt = document.ICO1.GetChannelCount()
        alert("Number of channels : " + cnt)
        for(i=0; i < cnt ; i++)
        {
            ChannelName = document.ICO1.GetChannelName(i)
            ChannelNum = document.ICO1.GetChannelNumber(ChannelName)
            alert("Index "+ i + " ChannelName " + ChannelName +
          " Number " + ChannelNum)
        }
    }
```

<contd...>

<contd...>

```
function GetGlobalChannelInfo(form)
    {
        var cnt
        var i
        var ChannelName
        var ChannelNum
        cnt = document.ICO1.GetGlobalChannelCount()
        alert(“Number of channels : ” + cnt)
        for(i=0; i < cnt ; i++)
        {
            ChannelName = document.ICO1.GetGlobalChannelName(i)
            ChannelNum = document.ICO1.GetGlobalChannelNumber(Channel-
Name)
            alert(“Index ”+ i + “ ChannelName ” + ChannelName +
           “ Number ” + ChannelNum)
        }
    }
    function GetMaxChannelInfo(form)
    {
        var cnt,cbRead,cbWrite
        cnt = document.ICO1.GetMaxChannelCount()
        cbRead = document.ICO1.GetMaxChannelRead()
        cbWrite = document.ICO1.GetMaxChannelWrite()
        alert(“Number of channels : ” + cnt + “ Max Read: ”+cbRead+
       “ Max Write: ”+cbWrite)
    }
    function GetChFlags(form)
    {
        var Flags
        Flags = document.ICO1.GetChannelFlags(form.SendChannel.value)
        alert(“Channel Flags: ” + Flags)
    }
    function SetChFlags(form)
    {
        var rc
        rc = document.ICO1.SetChannelFlags(form.SendChannel.value, 1)
        alert(“Channel Flags rc: ” + rc)
    }
//
    // GetRandomString
    //
    // Use random strings to stress the transmission of data to server
    // otherwise use compression to reduce the amount of data sent
    //
```

<contd...>

<contd…>

```
function GetRandomString(Length)
    {
    var RndStr
        var RndNum
        var ValidChars   =
"!#$%&'()*+,-./01234567890:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcde
fghijklmnopqrstuvwxyz{|}~"
        var cbValidChars = ValidChars.length

    RndStr = ""

    for(i = 0; i < Length; i++)
    {
            RndNum = GetRandomNumber(0,cbValidChars-1)
      RndStr += ValidChars.substring(RndNum, RndNum+1)
    }

    return(RndStr)
    }

    function GetRandomNumber(MinNum,MaxNum)
    {
    var RndNum

    if((MinNum < 0) || (MaxNum <0) || (MaxNum <= MinNum))
    {
      return(0)
    }

    RndNum = MinNum + Math.round(Math.random()*(MaxNum-MinNum))

    return(RndNum)
}

    function SendRandomString(ChannelName)
    {
        var cbMax
        var cbLen
        var RndStr
        var rc
```

<contd…>

<contd…>

```
cbMax = document.ICO1.GetMaxChannelWrite()
        cbLen = GetRandomNumber(1,cbMax)
        RndStr = GetRandomString(cbLen)
       rc = document.ICO1.SendChannelData(ChannelName, RndStr, cbLen,
2)
        return(rc)
    }
    function MyStartup()
    {
        document.ICO1.Startup()
    }
    //-->
   </script>
</head>
<body onLoad=“MyStartup()”>
<h1>Examples of Virtual Channel methods</h1>

<center>
<br>
<object
  id=“ICO1”
  classid=“clsid:238F6F83-B8B4-11cf-8771-00A024541EE3”
  height=600 width=800>
    <param name=“Address”       value=“192.1.1.128”>
    <param name=“VirtualChannels” value=“TEST1, TEST2”>
</object>
<br>
<form name=results>
<table>
<tr><td><input type=“button” value=“SendChannelData”
onClick=“WriteData(this.form)”></td>
    <td><input type=“button” value=“SendChannelData (random string)”
onClick=“SendRandomString(this.form.SendChannel.value)”></td></tr>
<tr><td><input type=“button” value=“Get Channel Info”
onClick=“GetChannelInfo(this.form)”></td>
    <td><input type=“button” value=“Get Global Channel Info”
onClick=“GetGlobalChannelInfo(this.form)”></td></tr>
<tr><td><input type=“button” value=“GetChannelFlags”
onClick=“GetChFlags(this.form)”></td>
    <td><input type=“button” value=“SetChannelFlags”
onClick=“SetChFlags(this.form)”></td></tr>
<tr><td><input type=“button” value=“Get Max Channel Info”
onClick=“GetMaxChannelInfo(this.form)”></td>
```

<contd…>

<contd…>

```
<tr><td>ChannelName:</td><td><input type=text name=SendChannel size=10
value="Test1"></td></tr>
<tr><td>ChannelData:</td><td><input type=text name=SendData size=50
value="abcdefghijklmnopqrstuvwxyz"></td></tr>
<tr><td>ChannelDataRead:</td><td><input type=text name=ReadData
size=60></td></tr>
<tr><td>LastError:</td><td><input type=text name=VCForm
size=5></td></tr>
<tr><td>LastClientError:</td><td><input type=text name=CltErr
size=5></td></tr>
<tr><td>ChannelName:</td><td><input type=text name=ChannelName
size=8></td></tr>
<tr><td>ChannelData:</td><td><input type=text name=ChannelData
size=60></td></tr>
<tr><td>BytesReceived:</td><td><input type=text name=BytesRecv
size=5></td></tr>
<tr><td>DataType:</td><td><input type=text name=DataType
size=5></td></tr>
<tr><td>PktCount:</td><td><input type=text name=PktCount size=5
value="0"></td></tr>
<tr><td>TotalPktSize:</td><td><input type=text name=TotalPktSize
size=10 value="0"></td></tr>

</table>
</form>
</center>

</body>
</html>
```

Automatic Application Resizing

This example illustrates the use of methods and properties applicable to automatic application resizing operations. It also demonstrates window calls to enable and disable keyboard and mouse input.

```
<!doctype html public “-//w3c//dtd html 4.0 transitional//en”>
<html>
<head>
  <meta http-equiv=“Content-Type” content=“text/html; charset=iso-
8859-1”>
  <title>Window call examples</title>
    <style type=“text/css”>
    #OBJ1 {position:absolute; left:100; top:100; width:640;
height:480;}
    #OBJ2 {position:absolute; left:100; top:100; width:640;
height:480;}
    #FORM {position:absolute; left:100; top:600; }
    </style>

    <script language=“javascript” for=“ICO1” event=“OnWindowSized(Wnd-
Type, Width,Height)”
    type=“text/javascript”>
    <!--
    ICO1_OnWindowSized(WndType, Width,Height);
    //-->
    </script>

    <script language=“javascript” for=“ICO1” event=“OnWindowMoved(Wnd-
Type, XPos,YPos)”
    type=“text/javascript”>
    <!--
    ICO1_OnWindowMoved(WndType, XPos,YPos);
    //-->
    </script>

    <script language=“javascript” for=“ICO1” event=“OnWindowCre-
ated(WndType, XPos,YPos,Width,Height)”
    type=“text/javascript”>
    <!--
    ICO1_OnWindowCreated(WndType, XPos,YPos,Width,Height);
    //-->
    </script>
```

<contd...>

<contd...>

```
<script language="javascript" for="ICO1" event="OnWindowDestroyed(Wnd-
Type)"
    type="text/javascript">
    <!--
    ICO1_OnWindowDestroyed(WndType);
    //-->
    </script>
    <script language="javascript" for="ICO1" event="OnWindowCloseRe-
quest()"
    type="text/javascript">
    <!--
    ICO1_OnWindowCloseRequest();
    //-->
    </script>
  <script language="JavaScript">
    <!--
    function MyStartup()
    {
        document.ICO1.Startup()
        GetSize(document.results)
        GetPosition(document.results)
        document.ICO1.SetProp("Title", "My session")
        document.ICO1.SetProp("ControlWindowText", "Undocked")
    }
    function SetPos(form)
    {
        document.ICO1.SetWindowPosition(parseInt(form.WndType.value),
parseInt(form.xPos.value), parseInt(form.yPos.value),
parseInt(form.WndFlags.value))
    }
    function SetSize(form)
    {
        document.ICO1.SetWindowSize(parseInt(form.WndType.value),
parseInt(form.Width.value), parseInt(form.Height.value),
parseInt(form.WndFlags.value))
    }
    function Show(form)
    {
        document.ICO1.DisplayWindow(parseInt(form.WndType.value))
    }
```

<contd...>

<contd...>

```
function Hide(form)
    {
        document.ICO1.HideWindow(parseInt(form.WndType.value))
    }
    function GetPosition(form)
    {
        form.xPos.value = document.ICO1.GetWindowXPosi-
tion(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
        form.yPos.value = document.ICO1.GetWindowYPosi-
tion(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
    }
    function GetSize(form)
    {
        form.Width.value = document.ICO1.GetWindow-
Width(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
        form.Height.value = document.ICO1.GetWin-
dowHeight(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
    }
    function Undock(form)
    {
        document.ICO1.UndockWindow()
    }
    function Dock(form)
    {
        document.ICO1.DockWindow()
    }
    function OnTop(form)
    {
        document.ICO1.PlaceWindowOnTop()
    }
    function OnBottom(form)
    {
        document.ICO1.PlaceWindowOnBottom()
}
    function Maximize(form)
    {
        document.ICO1.MaximizeWindow()
    }
```

<contd...>

<contd...>

```
function Minimize(form)
    {
        document.ICO1.MinimizeWindow()
    }
    function Restore(form)
    {
        document.ICO1.RestoreWindow()
    }
    function ScaleToFit(form)
    {
        document.ICO1.ScaleToFit()
    }
    function GetDesktopInfo(form)
    {
        form.Width.value = document.ICO1.GetSessionWidth()
        form.Height.value = document.ICO1.GetSessionHeight()
        form.Color.value = document.ICO1.GetSessionColorDepth()
    }
    function GetScreenInfo(form)
    {
        form.Width.value = document.ICO1.GetScreenWidth()
        form.Height.value = document.ICO1.GetScreenHeight()
        form.Color.value = document.ICO1.GetScreenColorDepth()
    }
    function ICO1_OnWindowSized(WndType, width, height)
    {
        document.results.WndType.value = WndType
        document.results.Width.value = width
        document.results.Height.value = height
    }
    function ICO1_OnWindowMoved(WndType, xPos, yPos)
    {
        document.results.WndType.value = WndType
        document.results.xPos.value = xPos
        document.results.yPos.value = yPos
    }
```

<contd...>

<contd...>

```
function ICO1_OnWindowCreated(WndType, xPos, yPos, Width, Height)
    {
        document.results.WndType.value = WndType
        document.results.xPos.value = xPos
        document.results.yPos.value = yPos
        document.results.Width.value = Width
        document.results.Height.value = Height
    }
function ICO1_OnWindowDestroyed(WndType)
    {
        document.results.WndType.value = WndType
    }

    function ICO1_OnWindowCloseRequest()
    {
        document.ICO1.DockWindow()
    }

//  -->
   </script>
</head>
<body onLoad="MyStartup()">
<h1>Window call examples</h1>

<center>
<div id=OBJ1>
<object
  id="ICO1"
  classid="clsid:238F6F83-B8B4-11cf-8771-00A024541EE3"
  width=640 height=480>
    <param name=application value="notepad">
    <param name=tcpbrowseraddress value=192.1.1.128>
</object>
</div>
<div id=OBJ2>
<img src=background.gif width=640 height=480></img>
</div>
</center>
<div id=FORM>
<form name=results>
<center>
<table>
<tr>
```

<contd...>

<contd…>

```
<td>WndType:</td>   <td><input type=text name=WndType size=5
value=0></td>
<td>WndFlags:</td>  <td><input type=text name=WndFlags size=5
value=0></td>
</tr>
<tr>
<td>Width:</td><td><input type=text name=Width size=5 value=640></td>
<td>Height:</td><td><input type=text name=Height size=5 value=480></
td>
</tr>
<tr>
<td>Color:</td>
<td><input type=text name=Color size=5 value=0>
</td>
</tr>
<tr>
<td>xPos:</td><td><input type=text name=xPos size=5 value=0></td>
<td>yPos:</td><td><input type=text name=yPos size=5 value=0></td>
</tr>
</table>
<table>
<tr>
<td align=center><input type="button" value="GetPosition"
onClick="GetPosition(this.form)"></td>
<td align=center><input type="button" value="SetPosition"
onClick="SetPos(this.form)"></td>
</tr>
<tr>
<td align=center><input type="button" value="GetSize" onClick="Get-
Size(this.form)"></td>
<td align=center><input type="button" value="SetSize" onClick="Set-
Size(this.form)"></td>
</tr>
<tr>
<td align=center><input type="button" value="Show"
onClick="Show(this.form)"></td>
<td align=center><input type="button" value="Hide"
onClick="Hide(this.form)"></td>
</tr>
```

<contd…>

<contd...>

```
<tr>
<td align=center><input type=“button” value=“Undock”
onClick=“Undock(this.form)”></td>
<td align=center><input type=“button” value=“Dock”
onClick=“Dock(this.form)”></td>
</tr>
<tr>
<td align=center><input type=“button” value=“Top”
onClick=“OnTop(this.form)”></td>
<td align=center><input type=“button” value=“Bottom” onClick=“OnBot-
tom(this.form)”></td>
</tr>
<tr>
<td align=center><input type=“button” value=“Minimize” onClick=“Mini-
mize(this.form)”></td>
<td align=center><input type=“button” value=“Maximize” onClick=“Maxi-
mize(this.form)”></td>
</tr>
<tr>
<td align=center><input type=“button” value=“Restore”
onClick=“Restore(this.form)”></td>
<td align=center><input type=“button” value=“ScaleToFit”
onClick=“ScaleToFit(this.form)”></td>
</tr>
<tr>
<td align=center><input type=“button” value=“AutoScale(ON)”
onClick=“document.ICO1.SetProp(‘AutoScale’,‘ON’)”></td>
<td align=center><input type=“button” value=“AutoScale(OFF)”
onClick=“document.ICO1.SetProp(‘AutoScale’,‘OFF’)”></td>
</tr>
<tr>
<td align=center><input type=“button” value=“ShowTitleBar”
onClick=“document.ICO1.ShowTitleBar()”></td>
<td align=center><input type=“button” value=“HideTitleBar”
onClick=“document.ICO1.HideTitleBar()”></td>
</tr>
<tr>
<td align=center><input type=“button” value=“EnableSizingBorder”
onClick=“document.ICO1.EnableSizingBorder()”></td>
<td align=center><input type=“button” value=“DisableSizingBorder”
onClick=“document.ICO1.DisableSizingBorder()”></td>
</tr>
```

<contd...>

<contd...>

```
<tr>
<td align=center><input type="button" value="FullScreenWindow"
onClick="document.ICO1.FullScreenWindow()"></td>
<td align=center><input type="button" value="FocusWindow"
onClick="document.ICO1.FocusWindow()"></td>
</tr>
<tr>
<td align=center><input type="button" value="AutoAppResize(ON)"
onClick="document.ICO1.SetProp('AutoAppResize', 'ON')"></td>
<td align=center><input type="button" value="AutoAppResize(OFF)"
onClick="document.ICO1.SetProp('AutoAppResize', 'OFF')"></td>
</tr>
<tr>
<td align=center><input type="button" value="GetDesktopInfo"
onClick="GetDesktopInfo(this.form)"></td>
<td align=center><input type="button" value="GetScreenInfo"
onClick="GetScreenInfo(this.form)"></td>
</tr>
<tr>
<td align=center><input type="button" value="NewWindow" onClick="docu-
ment.ICO1.NewWindow(0,0,0,0,0)"></td>
<td align=center><input type="button" value="DeleteWindow"
onClick="document.ICO1.DeleteWindow()"></td>
</tr>
<tr>
<td align=center><input type="button" value="EnableKeyboard"
onClick="document.ICO1.EnableKeyboardInput()"></td>
<td align=center><input type="button" value="DisableKeyboard"
onClick="document.ICO1.DisableKeyboardInput()"></td>
</tr>
<tr>
<td align=center><input type="button" value="EnableMouse"
onClick="document.ICO1.EnableMouseInput()"></td>
<td align=center><input type="button" value="DisableMouse"
onClick="document.ICO1.DisableMouseInput()"></td>
</tr>
</table>
</form>
</div>
</center>
</body>
</html>
```

Client Network Name and Address

This example illustrates the use of methods and properties available to get client network name and address information.

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
  <title>Client address example</title>
  <script language="JavaScript">
    <!--
    function GetClientNetworkName(form)
    {
        form.netname.value = document.ICO1.GetClientNetworkName()
    }
    function GetAddrCnt(form)
    {
        form.addrcnt.value = document.ICO1.GetClientAddressCount()
    }
    function GetAddr(form)
    {
        form.addr.value = document.ICO1.GetClientAddress(0)
    }
    function MyStartup()
    {
        document.ICO1.Startup()
    }
    //-->
   </script>
</head>
<body onLoad="MyStartup()">
<h1>Client address example</h1>
<center>
<br>
<object
  id="ICO1"
  classid="clsid:238F6F83-B8B4-11cf-8771-00A024541EE3"
  height=480 width=640>
<param name="Address" value="192.1.1.128">
</object>
<br>
</center>
```

<contd...>

<contd...>

```
<form name=results>
<center>
<table>
<tr><td>NetworkName:</td><td><input type=text name=netname
size=50></td></tr>
<tr><td>AddressCount:</td><td><input type=text name=addrcnt
size=5></td></tr>
<tr><td>Address:</td><td><input type=text name=addr size=50></td></tr>
</table>

<input type=button value="Get network name"
onClick="GetClientNetworkName(this.form)"><br>
<input type=button value="Get address count"
onClick="GetAddrCnt(this.form)"><br>
<input type=button value="Get address"
onClick="GetAddr(this.form)"><br>
</center>
</form>

</body>
</html>
```

Error Message Text from Error Codes

This example illustrates the use of methods available to extract error message text if the ICA Client Object error codes are specified.

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
  <title>Error messages example</title>
  <script language="JavaScript">
    <!--
    var hndWnd
    function MyStartup()
    {
        document.ICO1.Startup()
    }
    function GetErr(form)
    {
        form.ErrorText.value =
document.ICO1.GetErrorMessage(parseInt(form.ErrorCode.value))
    }
    function GetClientErr(form)
    {
        form.ErrorText.value =
document.ICO1.GetClientErrorMessage(parseInt(form.ErrorCode.value))
    }
//  -->
   </script>
</head>
<body onLoad="MyStartup()">
<h1>Error messages example</h1>
<center>
<br>
<object
  id="ICO1"
  classid="clsid:238F6F83-B8B4-11cf-8771-00A024541EE3"
  height=200 width=320>
    <param name="Address"   value="192.1.1.128">
</object>
<br>
</center>
<form name=results>
<center>
```

<contd...>

<contd...>

```
<table>
<tr><td>Error Code:</td><td><input type=text size=5  name="ErrorCode"
value=0></td></tr>
<tr><td>Error Text:</td><td><input type=text size=80
name="ErrorText"></td></tr>
<table>
<tr>
<td align=center><input type="button" value="GetErrorMessage"
onClick="GetErr(this.form)"></td>
<td align=center><input type="button" value="GetClientErrorMessage"
onClick="GetClientErr(this.form)"></td>
</tr>
</table>

</form>
</center>
</body>
</html>
```

Global Logoff and Disconnect

This example illustrates the use of methods available to log off or disconnect a group of sessions.

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html> <head>
      <meta http-equiv="Content-Type" content="text/html; char-
set=iso-8859-1">
      <title>Group Logoff/Disconnect example</title>
      <script language="javascript" for="ICO1" event="OnConnect()"
type="text/javascript">
      <!- -
          ICO1_OnConnect();
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnDiscon-
nect()" type="text/javascript">
      <!--
          ICO1_OnDisconnect();
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnConnect-
Failed()" type="text/javascript">
      <!--
          ICO1_OnConnectFailed();
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnDisconnect-
Failed()" type="text/javascript">
      <!--
          ICO1_OnDisconnectFailed();
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnDisconnect-
Sessions(hCommand)" type="text/javascript">
      <!--
          ICO1_OnDisconnectSessions(hCommand);
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnDisconnect-
SessionsFailed(hCommand)" type="text/javascript">
      <!--
          ICO1_OnDisconnectSessionsFailed(hCommand);
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnLogoffSes-
sions(hCommand)" type="text/javascript">
```

<contd…>

<contd...>

```
      <!--
          ICO1_OnLogoffSessions(hCommand);
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnLogoffSes-
sionsFailed(hCommand)" type="text/javascript">
      <!--
          ICO1_OnLogoffSessionsFailed(hCommand);
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnSession-
Switch(hOldSession, hNewSession)" type="text/javascript">
      <!--
          ICO1_OnSessionSwitch(hOldSession, hNewSession);
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnSessionEvent-
Pending(hSession, iEvent)" type="text/javascript">
      <!--
          ICO1_OnSessionEventPending(hSession, iEvent);
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnLogon()"
type="text/javascript">
      <!--
          ICO1_OnLogon();
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnPub-
lishedApp()" type="text/javascript">
      <!--
          ICO1_OnPublishedApp();
      //-->
      </script>
      <script language="javascript" for="ICO1" event="OnPublishedApp-
Failed()" type="text/javascript">
      <!--
          ICO1_OnPublishedAppFailed();
      //-->
      </script>
      <script language="JavaScript"><!--
          function ICO1_OnConnect()
          {
              alert("Connected")
          }
          function ICO1_OnDisconnect()
```

<contd...>

<contd...>

```
            {
                alert("Disconnected")
            }
            function ICO1_OnConnectFailed()
            {
                alert("Connect failed")
            }
            function ICO1_OnDisconnectFailed()
            {
                alert("Disconnect failed")
            }
            function ICO1_OnLogon()
            {
                alert("Logged in")
            }
            function ICO1_OnPublishedApp()
            {
                alert("Application launched")
            }
            function ICO1_OnPublishedAppFailed()
            {
                alert("Application launch failed")
            }
            function ICO1_OnDisconnectSessions(hCommand)
            {
                alert("OnDisconnectSessions " + hCommand)
                document.GlobalLogoff.DisconnectSessionshCmd.value = 
hCommand
                document.GlobalLogoff.DisconnectSessionsRC.value = 
ICO1.GetLastError()document.GlobalLogoff.DisconnectSessionsClien-
tRC.value = ICO1.GetLastClientError()
            }
            function ICO1_OnDisconnectSessionsFailed(hCommand)
            {
                alert("OnDisconnectSessionsFailed " + hCommand)
                document.GlobalLogoff.DisconnectSessionshCmd.value = 
hCommand
                document.GlobalLogoff.DisconnectSessionsRC.value = 
ICO1.GetLastError()
document.GlobalLogoff.LogoffSessionsClientRC.value = ICO1.GetLastCli-
entError()
            }
            function ICO1_OnLogoffSessions(hCommand)
            {
                alert("OnLogoffSessions " + hCommand)
                document.GlobalLogoff.LogoffSessionshCmd.value = hCom-
mand
```

<contd...>

<contd…>

```
            document.GlobalLogoff.LogoffSessionsRC.value = 
ICO1.GetLastError()
            document.GlobalLogoff.LogoffSessionsClientRC.value = 
ICO1.GetLastClientError()
        }
        function ICO1_OnLogoffSessionsFailed(hCommand)
        {
            alert("OnLogoffSessionsFailed " + hCommand)
            document.GlobalLogoff.LogoffSessionshCmd.value = hCom-
mand
            document.GlobalLogoff.LogoffSessionsRC.value = 
ICO1.GetLastError()
            document.GlobalLogoff.LogoffSessionsClientRC.value = 
ICO1.GetLastClientError()
        }
        function ICO1_OnSessionSwitch(hOldSession, hNewSession)
        {
            alert("OnSessionSwitch Old Session:" + hOldSession + " 
New Session: " + hNewSession)
        }
        function ICO1_OnSessionEventPending(hSession, iEvent)
        {
            alert("OnSessionEventPending Session:" + hSession + " 
Event: " + iEvent)
        }
        function DisconnectSessions(form)
        {
            form.DisconnectSessionshCmd.value = ICO1.DisconnectSes-
sions(form.DisconnectGroupId.value)
            form.DisconnectSessionsRC.value = ICO1.GetLastError()
            form.DisconnectSessionsClientRC.value = ICO1.GetLast-
ClientError()
        }
        function LogoffSessions(form)
        {
            form.LogoffSessionshCmd.value = ICO1.LogoffSes-
sions(form.LogoffGroupId.value)
            form.LogoffSessionsRC.value = ICO1.GetLastError()
            form.LogoffSessionsClientRC.value = ICO1.GetLastClien-
tError()
        }
        function SetSessionGroupId(form)
        {
            form.SetSessionGroupIdRC.value = ICO1.SetSession-
GroupId(form.SetGroupId.value)
            form.SetSessionGroupIdClientRC.value = ICO1.GetLastCli-
entError()
        }
```

<contd…>

<contd...>

```
          function SwitchSession(form)
          {
              form.SwitchRC.value       = ICO1.SwitchSes-
sion(form.SwitchhSession.value)
              form.SwitchClientRC.value = ICO1.GetLastClientError()
          }
          function GetSessionHandle(form)
          {
              form.GetSessionHandlehSession.value = ICO1.GetSession-
Handle()
              form.GetSessionHandleRC.value       = ICO1.GetLastEr-
ror()
              form.GetSessionHandleClientRC.value = ICO1.GetLastCli-
entError()
          }
          function GetSessionCount(form)
          {
              form.GetSessionCountNumber.value    = ICO1.GetSession-
Count()
              form.GetSessionCountRC.value        = ICO1.GetLastEr-
ror()
              form.GetSessionCountClientRC.value  = ICO1.GetLastCli-
entError()
          }
          function GetSessionHandleByIndex(form)
          {
              form.GSHBI_hSession.value  = ICO1.GetSessionHandleByIn-
dex(form.GSHBI_Index.value)
              form.GSHBI_RC.value         = ICO1.GetLastError()
              form.GSHBI_ClientRC.value  = ICO1.GetLastClientError()
          }
          function GetSessionGroupCount(form)
          {
              form.GSGC_Number.value    = ICO1.GetSessionGroup-
Count(form.GSGC_GroupId.value)
              form.GSGC_RC.value        = ICO1.GetLastError()
              form.GSGC_ClientRC.value  = ICO1.GetLastClientError()
          }
          function SetAddress(form)
          {
              ICO1.SetProp("Address", form.Address.value)
          }
          function SetSessionIdProp(form)
          {
              ICO1.SetProp("SessionId", form.SessionId.value)
          }
          function SetSessionGroupIdProp(form)
```

<contd...>

<contd...>

```
          {
              ICO1.SetProp("SessionGroupId", form.SessionGroupId-
Prop.value)
          }
          function SetSessionExitTimeoutProp(form)
          {
              ICO1.SetProp("SessionExitTimeout", form.SessionExitTim-
eoutProp.value)
          }
          function SetLaunchProp(form)
          {
              // enable or disable launching for next started session
              if(form.LaunchOption.checked)
              {
                  ICO1.SetProp("Launch", "TRUE")
              }
              else
              {
                  ICO1.SetProp("Launch", "FALSE")
              }
          }
          function SetDesiredHResProp(form)
          {
              ICO1.SetProp("DesiredHRes", form.DesiredHResProp.value)
          }
          function SetDesiredVResProp(form)
          {
              ICO1.SetProp("DesiredVRes", form.DesiredVResProp.value)
          }
          function SetTitleProp(form)
          {
              ICO1.SetProp("Title", form.TitleProp.value)
          }
          function SetESSCProp(form)
          }
          {
              var Value
              // enable or disable session sharing client support
              if(form.ESSCOption.checked)
              {
                  Value = "TRUE"
              }
              else
              {
                  Value = "FALSE"
              }
```

<contd...>

<*contd…*>

```
                ICO1.SetProp("EnableSessionSharingClient", Value)
            }
            function SetESSHProp(form)
            {
                var Value
                // enable or disable session sharing host support
                if(form.ESSHOption.checked)
                {
                    Value = "TRUE"
                }
                else
                {
                    Value = "FALSE"
                }
                ICO1.SetProp("EnableSessionSharingHost", Value)
            }
            function SetSSNProp(form)
            {
                ICO1.SetProp("SessionSharingName", form.SSNProp.value)
            }
            function SetSSLOProp(form)
            {
                var Value
                // enable or disable session sharing launching
                if(form.SSLOOption.checked)
                {
                    Value = "TRUE"
                }
                else
                {
                    Value = "FALSE"
                }
                ICO1.SetProp("SessionSharingLaunchOnly", Value)
            }
            function SetInitialProgramProp(form)
            {
                ICO1.SetProp("InitialProgram", form.InitialProgram-
Prop.value)
            }
            function SetLCLProp(form)
            {
                ICO1.SetProp("LongCommandLine", form.LCLProp.value)
            }
            function SetTWIModeProp(form)
            {
                var Value

                // enable or disable seamless mode
```

<*contd…*>

<contd...>

```
            if(form.TWIModeOption.checked)
            {
                Value = "TRUE"
            }
            else
            {
                Value = "FALSE"
            }
            ICO1.SetProp("TWIMode", Value)
        }
        function SetTWIDSSProp(form)
        {
            var Value
            // enable or disable session sharing with Seamless ses-
sions
            if(form.TWIDSSOption.checked)
            {
                Value = "TRUE"
            }
            else
            {
                Value = "FALSE"
            }
            ICO1.SetProp("TWIDisableSessionSharing", Value)
        }
        function MyStartup()
        {
            document.ICO1.Startup()

            // initialize all the settings
            SetAddress(document.GlobalLogoff)
            SetSessionIdProp(document.GlobalLogoff)
            SetSessionGroupIdProp(document.GlobalLogoff)
            SetSessionExitTimeoutProp(document.GlobalLogoff)
            SetLaunchProp(document.GlobalLogoff)
            SetESSCProp(document.GlobalLogoff)
            SetESSHProp(document.GlobalLogoff)
            SetSSLOProp(document.GlobalLogoff)
            SetDesiredHResProp(document.GlobalLogoff)
            SetDesiredVResProp(document.GlobalLogoff)
            SetTitleProp(document.GlobalLogoff)
            SetSSNProp(document.GlobalLogoff)
            SetInitialProgramProp(document.GlobalLogoff)
            SetLCLProp(document.GlobalLogoff)
            SetTWIModeProp(document.GlobalLogoff)
            SetTWIDSSProp(document.GlobalLogoff)
        }
    //  -->      }
    </script>
```

<contd...>

<contd...>

```
   </head>
   <body onLoad="MyStartup()">
       <h1>Group Logoff/Disconnect example</h1>
       <center>
           <object id="ICO1" classid="clsid:238F6F83-B8B4-11cf-8771-
00A024541EE3" height="482" width="642" VIEWASTEXT>
               <param name="SessionId" value="MyId">
               <param name="SessionGroupId" value="MyGroupId">
           </object>
       </center>
       <center>
       <form name="GlobalLogoff">
               <table border=1 cellpadding="3">
                   <tr>
                       <th>Property</th>
                       <th>Value</th>
                       <th>Set</th>
                   </tr>
                   <tr>
                       <td>Address</td>
                       <td><input type="text" name="Address" size="20"
value="192.1.1.128"></td>
                       <td><input type="button" value="Set Address"
onClick="SetAddress(this.form)"></td>
                   </tr>
                   <tr>
                       <td>SessionId</td>
                       <td><input type="text" name="SessionId"
size="20" value="MyId"></td>
                       <td><input type="button" value="Set SessionId"
onClick="SetSessionIdProp(this.form)">
</td>
                   </tr>
                   <tr>
                       <td>SessionGroupId</td>
                       <td><input type="text" name="SessionGroupId-
Prop" size="20" value="MyGroupId"></td>
                       <td><input type="button" value="Set Session-
GroupId" onClick="SetSessionGroupIdProp(this.form)"></td>
                   </tr>
                   <tr>
                       <td>SessionExitTimeout</td>
                       <td><input type="text" name="SessionExitTime-
outProp" size="5" value="60">
</td>
```

<contd...>

<contd…>

```
<td><input type="button" value="Set SessionExitTimeout" onClick="Set-
SessionExitTimeoutProp(this.form)">
</td>
                   </tr>
                   <tr>
                       <td>Launch</td>
                       <td><input type="checkbox" name="LaunchOption" 
onClick="SetLaunchProp(this.form)"></td>
                       <td>&nbsp;</td>
                   </tr>
                   <tr>
                       <td>EnableSessionSharingClient</td>
                       <td><input type="checkbox" name="ESSCOption" 
onClick="SetESSCProp(this.form)"></td>
                       <td>&nbsp;</td>
                   </tr>
                   <tr>
                       <td>EnableSessionSharingHost</td>
                       <td><input type="checkbox" name="ESSHOption" 
onClick="SetESSHProp(this.form)"></td>
                       <td>&nbsp;</td>
                   </tr>
                   <tr>
                       <td>SessionSharingLaunchOnly</td>
                       <td><input type="checkbox" name="SSLOOption" 
onClick="SetSSLOProp(this.form)"></td>
                       <td>&nbsp;</td>
                   </tr>
                   <tr>
                       <td>TWIMode</td>
                       <td><input type="checkbox" name="TWIModeOp-
tion" onClick="SetTWIModeProp(this.form)"></td>
                       <td>&nbsp;</td>
                   </tr>
                   <tr>
                       <td>TWIDisableSessionSharing</td>
                       <td><input type="checkbox" name="TWIDSSOption" 
onClick="SetTWIDSSProp(this.form)"></td>
                       <td>&nbsp;</td>
                   </tr>
                   <tr>
                       <td>DesiredHRes</td>
                       <td><input type="text" name="DesiredHResProp" 
size="5" value="640"></td>
```

<contd…>

<contd...>

```
                        <td><input type="button" value="Set 
DesiredHRes" onClick="SetDesiredHResProp(this.form)"></td>
                    </tr>
                    <tr>
                        <td>DesiredVRes</td>
                        <td><input type="text" name="DesiredVResProp" 
size="5" value="480"></td>
                        <td><input type="button" value="Set Desired-
VRes" onClick="SetDesiredVResProp(this.form)"></td>
                    </tr>
                    <tr>
                        <td>Title</td>
                        <td><input type="text" name="TitleProp" 
size="20" value="My title"></td>
                        <td><input type="button" value="Set Title" 
onClick="SetTitleProp(this.form)"></td>
                    </tr>
                    <tr>
                        <td>SessionSharingName</td>
                        <td><input type="text" name="SSNProp" size="20" 
value="SessionShare"></td>
                        <td><input type="button" value="Set Session-
SharingName" onClick="SetSSNProp(this.form)"></td>
                    </tr>
                    <tr>
                        <td>InitialProgram</td>
                        <td><input type="text" name="InitialProgram-
Prop" size="20" value="#Notepad"></td>
                        <td><input type="button" value="Set InitialPro-
gram" onClick="SetInitialProgramProp(this.form)"></td>
                    </tr>
                    <tr>
                        <td>LongCommandLine</td>
                        <td><input type="text" name="LCLProp" size="20" 
value=""></td>
                        <td><input type="button" value="Set LongCom-
mandLine" onClick="SetLCLProp(this.form)"></td>
                    </tr>
                    <tr>
                        <th>Method</th>
                        <th>Parameters</th>
                        <th>Run</th>
                    </tr>
                    <tr>
                        <td>Connect</td>
                        <td>&nbsp;</td>
                        <td><input type="button" value="Connect" 
onClick="ICO1.Connect()"></td>
                    </tr>
```

<contd...>

<contd...>

```
                        <tr>
                        <td>Disconnect</td>
                        <td>&nbsp;</td>
                        <td><input type="button" value="Disconnect"
onClick="ICO1.Disconnect()"></td>
                    </tr>
                    <tr>
                        <td>Logoff</td>
                        <td>&nbsp;</td>
                        <td><input type="button" value="Logoff"
onClick="ICO1.Logoff()"></td>
                    </tr>
                    <tr>
                        <td>AttachSession</td>
                        <td>&nbsp;</td>
                        <td><input type="button" value="AttachSession"
onClick="ICO1.AttachSession(this.form.SessionId.value)"></td>
                    </tr>
                    <tr>
                        <td>DetachSession</td>
                        <td>&nbsp;</td>
                        <td><input type="button" value="DetachSession"
onClick="ICO1.DetachSession(this.form.SessionId.value)"></td>
                    </tr>
                </table>
                <p>
                <p>
                    <table border=1 cellpadding="3">
                        <tr>
                            <th>Function</th>
                            <th>Parameters</th>
                            <th>Run</th>
                            <th>hCommand</th>
                            <th>ICO Result</th>
                            <th>Client Result</th>
                        </tr>
                        <tr>
                            <td>DisconnectSessions</td>
                            <td>GroupId:<input type="text" name="Dis-
connectGroupId" size="20" value="MyGroupId"></td>
                            <td><input type="button" value="Disconnect-
Sessions" onClick="DisconnectSessions(this.form)"></td>
```

<contd...>

<contd…>

```
      <td><input type="text" size="5" name="DisconnectSessionsh-
Cmd"></td>
                            <td><input type="text" size="5" name="Dis-
connectSessionsRC"></td>
                            <td><input type="text" size="5" name="Dis-
connectSessionsClientRC"></td>
                        </tr>
</td>
                        <tr>
                            <td>LogoffSessions</td>
                            <td>GroupId:<input type="text"
name="LogoffGroupId" size="20" value="MyGroupId">
</td>
                            <td><input type="button" value="LogoffSes-
sions" onClick="LogoffSessions(this.form)"></td>
                            <td><input type="text" size="5"
name="LogoffSessionshCmd"></td>
                            <td><input type="text" size="5"
name="LogoffSessionsRC"></td>
                            <td><input type="text" size="5"
name="LogoffSessionsClientRC"></td>
                        </tr>
                        <tr>
                            <th>Function</th>
                            <th>Parameters</th>
                            <th>Run</th>
                            <th>ICO Result</th>
                            <th>Client Result</th>
                        </tr>
                        <tr>
                            <td>SetSessionGroupId</td>
                            <td>GroupId:<input type="text" name="Set-
GroupId" size="20"></td>
                            <td><input type="button" value="SetSession-
GroupId" onClick="SetSessionGroupId(this.form)"></td>
                            <td><input type="text" name="SetSession-
GroupIdRC" size="5"></td>
                            <td><input type="text" name="SetSession-
GroupIdClientRC" size="5"></td>
                        </tr>
                        <tr>
                            <td>SwitchSession</td>
                            <td>hSession:<input type="text"
name="SwitchhSession" value="0" size="5"></td>
                            <td><input type="button" value="SwitchSes-
sion" onClick="SwitchSession(this.form)"></td>
```

<contd…>

<contd…>

```
              <td><input type="text" name="SwitchRC" size="5">
</td>
<td><input type="text" name="SwitchClientRC" size="5"></td>
</tr>
              <tr>
                  <th>Function</th>
                  <th>Parameters</th>
                  <th>Run</th>
                  <th>hSession</th>
                  <th>ICO Result</th>
                  <th>Client Result</th>
              </tr>
              <tr>
                  <td>GetSessionHandle</td>
                  <td>&nbsp;</td>
                  <td><input type="button" value="GetSessionHandle"
onClick="GetSessionHandle(this.form)"></td>
                  <td><input type="text" name="GetSessionHandlehSes-
sion" size="5</td">
                  <td><input type="text" name="GetSessionHandleRC"
size="5"></td>
                  <td><input type="text" name="GetSessionHandleClien-
tRC" size="5"></td>
              </tr>
              <tr>
                  td>GetSessionHandleByIndex</td>
                  <td>Index: <input type="text" name="GSHBI_Index"
size="5" value="0"></td>
                  <td><input type="button" value="GetSessionHandleBy-
Index" onClick="GetSessionHandleByIndex(this.form)"></td>
                  <td><input type="text" name="GSHBI_hSession"
size="5</td">
                  <td><input type="text" name="GSHBI_RC" size="5"></
td>
                  <td><input type="text" name="GSHBI_ClientRC"
size="5"></td>
                  </tr>
              <tr>
                  <th>Function</th>
                  <th>Parameters</th>
                  <th>Run</th>
                  <th>Count</th>
                  <th>ICO Result</th>
                  <th>Client Result</th>
              </tr>
```

<contd…>

<contd…>

```
<tr>
                   <td>GetSessionCount</td>
                   <td>&nbsp;</td>
                   <td><input type="button" value="GetSessionCount"
onClick="GetSessionCount(this.form)"></td>
                   <td><input type="text" name="GetSessionCountNum-
ber" size="5</td">
                   <td><input type="text" name="GetSessionCountRC"
size="5"></td>
                   <td><input type="text" name="GetSessionCountClien-
tRC" size="5"></td>
               </tr>
               <tr>
                   <td>GetSessionGroupCount</td>
                   <td>GroupId:<input type="text" name="GSGC_GroupId"
size="20" value="MyGroupId">
</td>
                   <td><input type="button" value="GetSessionGroup-
Count" onClick="GetSessionGroupCount(this.form)"></td>
                   <td><input type="text" name="GSGC_Number" size="5</
td">
                   <td><input type="text" name="GSGC_RC" size="5"></
td>
                   <td><input type="text" name="GSGC_ClientRC"
size="5"></td>
               </tr>
               </table>
       </form>
       </center>
   </body>
</html>
```

Error Codes

This chapter contains the complete list of error codes that can be returned when an error occurs during ICA connection negotiation or after an ICA connection is established. The ICA Client Object logs details of the last ICA Client Object and client errors that occurred as a result of the last request.

You can identify the reason the connection failed or was disconnected by calling the event handler to get values for GetLastError and GetLastClientError.

This chapter contains the following error codes:

- “ICA Client Object Error Codes”
- “Client Error Codes”

ICA Client Object Error Codes

An ICA Client Object error is the last error reported for the ICA Client Object. The ICA Client Object has well-defined error codes within specific ranges of valid values. Error values are made available externally by the two error handling methods, GetLastError and GetLastClientError, respectively.

Error Number	Defined Name	Ver.	Error String
0	ICO_NO_ERROR	2.0	“No Error”
1	ICO_ERROR	2.0	“Generic Error”
2	ICO_ERROR_NO_MEMORY	2.0	“No memory available”
3	ICO_ERROR_TOO_MANY_INSTANCES	2.0	“Cannot start more than one ICA connection. Please close previous connection.”
4	ICO_ERROR_TIMEOUT	2.0	“Timeout error”
5	ICO_ERROR_INITIALIZING	2.0	“Error initializing ICA Client Object”
6	ICO_ERROR_DELETING	2.0	“Error deleting ICA Client Object”
7	ICO_ERROR_INVALID_SESSIONID	2.2	“Invalid SessionId specified”
8	ICO_ERROR_INVALID_TIMEOUT	2.2	“Invalid time-out specified”
9	ICO_ERROR_INVALID_LIMIT	2.2	“Invalid cache limit specified”
10	ICO_ERROR_INVALID_PARAMETER	2.0	“Invalid parameter”
11	ICO_ERROR_INVALID_HANDLE	2.0	“Invalid handle”
12	ICO_ERROR_INVALID_STATETYPE	2.0	“Invalid state type”
13	ICO_ERROR_UNSUPPORTED_FUNCTION	2.0	“Unsupported function”
14	ICO_ERROR_INVALID_WINDOW	2.0	“Invalid window”
15	ICO_ERROR_INVALID_INDEX	2.0	“Invalid index parameter”
16	ICO_ERROR_INVALID_POINTER	2.0	“Invalid pointer”
17	ICO_ERROR_INVALID_TYPE	2.0	“Invalid type”
18	ICO_ERROR_INVALID_OPERATION	2.0	“Invalid operation”
19	ICO_ERROR_OBSOLETE	2.0	“Obsolete”
20	ICO_ERROR_NO_FILENAME	2.0	“No filename specified”
21	ICO_ERROR_BUILDING_FILENAME	2.0	“Error building filename”
22	ICO_ERROR_CREATING_FILE	2.0	“Error creating file”
23	ICO_ERROR_OPENING_FILE	2.0	“Error opening file”

Error Number	Defined Name	Ver.	Error String
24	ICO_ERROR_DELETING_FILE	2.0	“Error deleting file”
25	ICO_ERROR_WRITING_FILE	2.0	“Error writing file”
26	ICO_ERROR_READING_FILE	2.0	“Error reading file”
27	ICO_ERROR_FINDING_FILE	2.0	“Error finding file”
28	ICO_ERROR_SETTING_FILEPOSITION	2.0	“Error setting file position”
29	ICO_ERROR_GETTING_BOOT_DRIVE	2.0	“Error getting boot drive location”
30	ICO_ERROR_COPYING_FILE	2.0	“Error copying file”
35	ICO_ERROR_MESSAGES_DISABLED	2.0	“Messages disabled”
36	ICO_ERROR_MESSAGE_TOO_LONG	2.0	“Message too long”
37	ICO_ERROR_LOADING_RESOURCE	2.0	“Error loading string resource”
38	ICO_ERROR_SETTING_STATE	2.0	“Error setting state flag”
39	ICO_ERROR_GETTING_STATE	2.0	“Error getting state flag”
40	ICO_ERROR_READONLY_STATETYPE	2.0	“Error setting read-only state flag”
41	ICO_ERROR_BUFFER_TOO_SMALL	2.0	“Buffer is too small”
42	ICO_ERROR_EMPTY_CLIENT_PATH	2.0	“Client path not found”
43	ICO_ERROR_GETTING_PROPERTY	2.0	“Error getting property”
44	ICO_ERROR_SETTING_PROPERTY	2.0	“Error setting property”
45	ICO_ERROR_CLEARING_PROPERTIES	2.0	“Error clearing properties”
46	ICO_ERROR_RESET_PROPERTIES	2.0	“Error resetting properties”
47	ICO_ERROR_PROPERTY_SNAPSHOT	2.0	“Error taking property snapshot”
48	ICO_ERROR_DELETING_PROPERTY	2.0	“Error deleting property”
49	ICO_ERROR_FINDING_PROPERTY	2.0	“Error finding property”
50	ICO_ERROR_OPENING_REG_KEY	2.0	“Error opening registry key”
51	ICO_ERROR_QUERYING_REG_VALUE	2.0	“Error querying registry key”
52	ICO_ERROR_NO_MORE_TIMERS	2.0	“No more timers”
53	ICO_ERROR_TIMERID_NOT_FOUND	2.0	“Error finding timer ID”
54	ICO_ERROR_CREATING_TIMER	2.0	“Error creating timer”
55	ICO_ERROR_FINDING_SECTION	2.0	“Error finding INI section”
56	ICO_ERROR_GETTING_SERVER_NAME	2.0	“Error getting server name”
57	ICO_ERROR_GETTING_VALUE	2.0	“Error getting value”

Error Number	Defined Name	Ver.	Error String
58	ICO_ERROR_DELIVERING_EVENT	2.0	“Error delivering event”
59	ICO_ERROR_NO_MORE_EVENTS	2.0	“No more events available”
60	ICO_ERROR_NO_CLIENT_WINDOW	2.0	“No client window is available”
61	ICO_ERROR_RUNNING_CLIENT	2.0	“Error running the client”
62	ICO_ERROR_ALREADY_STARTED	2.0	“Client is already started”
63	ICO_ERROR_NO_WINDOW	2.0	“No window is available”
64	ICO_ERROR_ICAFILE_NOT_FOUND	2.0	“The ICA file could not be found.”
65	ICO_ERROR_ALREADY_CONNECTED	2.0	“Client is already connected”
66	ICO_ERROR_NOT_CONNECTED	2.0	“Client is not connected”
67	ICO_ERROR_LOADING_ENGINE	2.0	“Error loading the client”
68	ICO_ERROR_UNLOADING_ENGINE	2.0	“Error unloading the client”
69	ICO_ERROR_OPENING_ENGINE	2.0	“Error opening communication to client”
70	ICO_ERROR_CONNECTING_TO_SERVER	2.0	“Error connecting to server”
71	ICO_ERROR_LOADING_SESSION	2.0	“Error loading client session”
72	ICO_ERROR_DISCONNECTING	2.0	“Error disconnecting from server”
73	ICO_ERROR_LOADING_DLL	2.0	“Error loading DLL”
74	ICO_ERROR_UNLOADING_DLL	2.0	“Error unloading DLL”
75	ICO_ERROR_LOADING_FUNCTION	2.0	“Error loading DLL function”
76	ICO_ERROR_UNLOADING_FUNCTION	2.0	“Error unloading DLL function”
77	ICO_ERROR_DLL_ALREADY_LOADED	2.0	“DLL is already loaded”
78	ICO_ERROR_DLL_NOT_LOADED	2.0	“DLL is not loaded”
79	ICO_ERROR_FUNCTION_NOT_LOADED	2.0	“Function is not loaded”
80	ICO_ERROR_LOGGING_OFF	2.0	“Error logging off”
81	ICO_ERROR_RUNNING_APP	2.0	“Error running published application”
82	ICO_ERROR_IPC_GET_INFO	2.0	“Error getting client information”
83	ICO_ERROR_SETTING_LOG_INFO	2.0	“Error setting log information”
84	ICO_ERROR_NO_CONNECT_ACTION	2.0	“No connect action was performed”
85	ICO_ERROR_NO_PROPERTIES	2.0	“No properties are available”
86	ICO_ERROR_NO_STREAM_DONE	2.0	“The ICA file could not be downloaded.”

Error Number	Defined Name	Ver.	Error String
87	ICO_ERROR_WINDOW_TOO_LARGE	2.0	“Window size is too large”
88	ICO_ERROR_NOT_STARTED	2.0	“Client is not started”
89	ICO_ERROR_GETTING_VERSION_INFO	2.0	“Error getting version information”
90	ICO_ERROR_LOGGING_STRING	2.0	“Error logging string to logfile”
91	ICO_ERROR_VERSION_TOO_OLD	2.0	“Client version is too old”
92	ICO_ERROR_GETTING_LAST_ERROR	2.0	“Error getting last client error”
93	ICO_ERROR_VALUE_ALREADY_SET	2.0	“Value already set”
94	ICO_ERROR_GETTING_ICAFILE_VALUE	2.0	“Error getting ICAFile value”
95	ICO_ERROR_NO_CONNECTION_ENTRY	2.0	“No connection entry specified”
96	ICO_ERROR_SCALING	2.1	“Error with scaling operation”
97	ICO_ERROR_CLIENT_CONNECTION	2.1	“Error with client connection to server”
98	ICO_ERROR_VIRTUAL_CHANNEL	2.2	“Error with virtual channel support”
99	ICO_ERROR_NAME_ENUMERATION	2.2	“Error with name enumeration support”
100	ICO_ERROR_NAME_RESOLUTION	2.2	“Error with name resolution support”
101	ICO_ERROR_SETTING_WINDOW_SIZE	2.2	“Error setting the window size”
102	ICO_ERROR_SETTING_WINDOW_POS	2.2	“Error setting the window position”
103	ICO_ERROR_UNDOCKING_WINDOW	2.2	“Error undocking the window”
104	ICO_ERROR_DOCKING_WINDOW	2.2	“Error docking the window”
105	ICO_ERROR_PLACING_WINDOW	2.2	“Error placing the window”
106	ICO_ERROR_WINDOW_IS_DOCKED	2.2	“Cannot perform operation on docked window”
107	ICO_ERROR_WINDOW_ALREADY_EXIST	2.2	“Cannot create window since it already exists”
108	ICO_ERROR_DRIVER_NOT_LOADED	2.2	“Driver is not loaded”
109	ICO_ERROR_RESIZING_APPLICATION	2.2	“Error resizing application window”
110	ICO_ERROR_CREATING_WINDOW	2.2	“Error creating window”
111	ICO_ERROR_GETTING_ICON	2.2	“Error getting icon from file”
112	ICO_ERROR_GETTING_DESKTOPINFO	2.2	“Error getting desktop information”
113	ICO_ERROR_GETTING_KEYBOARD_STATE	2.2	“Error getting the keyboard state”
114	ICO_ERROR_SETTING_KEYBOARD_STATE	2.2	“Error setting the keyboard state”
115	ICO_ERROR_GETTING_MOUSE_STATE	2.2	“Error getting the mouse state”

Error Number	Defined Name	Ver.	Error String
116	ICO_ERROR_SETTING_MOUSE_STATE	2.2	"Error setting the mouse state"
117	ICO_ERROR_ATTACHING_SESSION	2.2	"Error attaching session"
118	ICO_ERROR_LOCKING CLIENT	2.2	"Error locking client for access"
119	ICO_ERROR_SESSION_CACHING_DISABLED	2.2	"Session caching feature is disabled"
120	ICO_ERROR_SESSIONID_NOT_FOUND	2.2	"Session ID was not found"
121	ICO_ERROR_GETTING_CLIENT_DATA	2.2	"Error getting data from the client"
122	ICO_ERROR_SETTING_CLIENT_DATA	2.2	"Error setting data in the client"
123	ICO_ERROR_CLOSING_CLIENT	2.2	"Error closing the client"
124	ICO_ERROR_SCALING_PASSTHRU_CLIENT	2.2	"Unable to perform scaling operation on pass-through client"
125	ICO_ERROR_SCALING_AUTOAPPRESIZE	2.3	"Unable to perform scaling operation with AutoAppResize enabled"
126	ICO_ERROR_UNMARSHALLING_INTERFACE	2.3	"Error getting the session interface"
127	ICO_ERROR_SESSION_EXIT_TIMEOUT	2.3	"Timeout processing session group logoff/disconnect"
128	ICO_ERROR_DISCONNECT_SESSION_FAILED	2.3	"Disconnecting a group of sessions has failed"
129	ICO_ERROR_INVALID_SESSION_HANDLE	2.3	"Session handle is invalid"
130	ICO_ERROR_LOGOFF_SESSION_FAILED	2.3	"Logging off a group of sessions has failed"
131	ICO_ERROR_NO_SESSIONS_IN_GROUP	2.3	"No sessions in the group"
132	ICO_ERROR_SESSION_ALREADY_SELECTED	2.3	"Session has already been selected"
133	ICO_ERROR_CANNOT_SCRIPT_IN_ZONE	2.3	"Security Alert: It is not safe to script the Citrix ICA Client Object in the Internet or Restricted Zones\n\n URL: %s \n\nIf this Web site is trusted, add it to the Trusted Zone list in Internet Explorer."

Client Error Codes

A client error is the last error obtained from the client engine. The client has well-defined error codes within specific ranges of valid values. Error values are made available externally by the two error handling methods, GetLastError and GetLastClientError, respectively.

Error Number	Defined Name	Version	Error String
0	CLIENT_STATUS_SUCCESS	2.1	“Completed successfully”
1000	CLIENT_ERROR	2.1	“Client error”
1001	CLIENT_ERROR_NO_MEMORY	2.1	“Insufficient memory”
1002	CLIENT_ERROR_BAD_OVERLAY	2.1	“Bad overlay”
1003	CLIENT_ERROR_BAD_PROCINDEX	2.1	“Bad procedure index”
1004	CLIENT_ERROR_BUFFER_TOO_SMALL	2.1	“Buffer too small”
1005	CLIENT_ERROR_CORRUPT_ALLOC_HEADER	2.1	“Corrupt memory header”
1006	CLIENT_ERROR_CORRUPT_ALLOC_TRAILER	2.1	“Corrupt memory trailer”
1007	CLIENT_ERROR_CORRUPT_FREE_HEADER	2.1	“Corrupt free header”
1008	CLIENT_ERROR_CORRUPT_SEG_HEADER	2.1	“Corrupt segment header”
1009	CLIENT_ERROR_MEM_ALREADY_FREE	2.1	“Memory already free”
1010	CLIENT_ERROR_MEM_TYPE_MISMATCH	2.1	“Memory type mismatch”
1011	CLIENT_ERROR_NULL_MEM_POINTER	2.1	“Null pointer”
1012	CLIENT_ERROR_IO_PENDING	2.1	“I/O pending”
1013	CLIENT_ERROR_INVALID_BUFFER_SIZE	2.1	“Invalid buffer size”
1014	CLIENT_ERROR_INVALID_MODE	2.1	“Invalid mode”
1015	CLIENT_ERROR_NOT_OPEN	2.1	“Not open”
1016	CLIENT_ERROR_NO_OUTBUF	2.1	“No output buffer”
1017	CLIENT_ERROR_DLL_PROCEDURE_NOT_FOUND	2.1	“DLL procedure not found”
1018	CLIENT_ERROR_DLL_LARGER_THAN_256K	2.1	“DLL larger than 256K”
1019	CLIENT_ERROR_DLL_BAD_EXEHEADER	2.1	“Corrupted DLL”
1020	CLIENT_ERROR_OPEN_FAILED	2.1	“Open failed”
1021	CLIENT_ERROR_INVALID_HANDLE	2.1	“Invalid handle”
1022	CLIENT_ERROR_VD_NOT_FOUND	2.1	“Virtual Driver not found”
1023	CLIENT_ERROR_WD_NAME_NOT_FOUND	2.1	“WinStation Driver name not found in MODULE.INI”

Error Number	Defined Name	Version	Error String
1024	CLIENT_ERROR_PD_NAME_NOT_FOUND	2.1	"Protocol Driver name not found in MODULE.INI"
1025	CLIENT_ERROR_THINWIRE_OUTOFSYNC	2.1	"Thinwire out of sync"
1026	CLIENT_ERROR_NO_MOUSE	2.1	"Mouse not available"
1027	CLIENT_ERROR_INVALID_CALL	2.1	"Invalid request"
1028	CLIENT_ERROR_QUEUE_FULL	2.1	"Queue is full"
1029	CLIENT_ERROR_INVALID_DLL_LOAD	2.1	"Invalid DLL load"
1030	CLIENT_ERROR_PD_ERROR	2.1	"Protocol Driver error"
1031	CLIENT_ERROR_VD_ERROR	2.1	"Virtual Driver error"
1032	CLIENT_ERROR_ALREADY_OPEN	2.1	"Already open"
1033	CLIENT_ERROR_PORT_NOT_AVAILABLE	2.1	"Port not available"
1034	CLIENT_ERROR_IO_ERROR	2.1	"Communication I/O Error"
1035	CLIENT_ERROR_THINWIRE_CACHE_ERROR	2.1	"Thinwire cache error"
1036	CLIENT_ERROR_BAD_FILE	2.1	"Bad file"
1037	CLIENT_ERROR_CONFIG_NOT_FOUND	2.1	"WFCLIENT.INI not found"
1038	CLIENT_ERROR_SERVER_FILE_NOT_FOUND	2.1	"APPSRV.INI not found"
1039	CLIENT_ERROR_PROTOCOL_FILE_NOT_FOUND	2.1	"MODULE.INI not found"
1040	CLIENT_ERROR_MODEM_FILE_NOT_FOUND	2.1	"MODEM.INI not found"
1041	CLIENT_ERROR_LAN_NOT_AVAILABLE	2.1	"LAN not available"
1042	CLIENT_ERROR_PD_TRANSPORT_UNAVAILABLE	2.1	"Transport not available"
1043	CLIENT_ERROR_INVALID_PARAMETER	2.1	"Invalid Parameter"
1044	CLIENT_ERROR_WD_NOT_LOADED	2.1	"No WinStation Driver is loaded"
1045	CLIENT_ERROR_NOT_CONNECTED	2.1	"Not connected to a Citrix Server"
1046	CLIENT_ERROR_VD_NOT_LOADED	2.1	"The Virtual Driver is not loaded"
1047	CLIENT_ERROR_ALREADY_CONNECTED	2.1	"Already connected to a Citrix Server"
1048	CLIENT_ERROR_WFENGINE_NOT_FOUND	2.1	"Cannot find the Citrix ICA Engine %s"
1049	CLIENT_ERROR_ENTRY_NOT_FOUND	2.1	"Entry not found in APPSRV.INI"

Error Number	Defined Name	Version	Error String
1050	CLIENT_ERROR_PD_ENTRY_NOT_FOUND	2.1	“Protocol Driver for Entry not found in APPSRV.INI”
1051	CLIENT_ERROR_WD_ENTRY_NOT_FOUND	2.1	“WinStation Driver for Entry not found in APPSRV.INI”
1052	CLIENT_ERROR_PD_SECTION_NOT_FOUND	2.1	“Protocol Driver section not found in MODULE.INI”
1053	CLIENT_ERROR_WD_SECTION_NOT_FOUND	2.1	“WinStation Driver section not found in MODULE.INI”
1054	CLIENT_ERROR_DEVICE_SECTION_NOT_FOUND	2.1	“Device section not found in WFCLIENT.INI”
1055	CLIENT_ERROR_VD_SECTION_NOT_FOUND	2.1	“Virtual Driver section not found in MODULE.INI”
1056	CLIENT_ERROR_VD_NAME_NOT_FOUND	2.1	“Virtual Driver name not found in MODULE.INI”
1057	CLIENT_ERROR_SERVER_CONFIG_NOT_FOUND	2.1	“[WFClient] section not found in APPSRV.INI”
1058	CLIENT_ERROR_CONFIG_CONFIG_NOT_FOUND	2.1	“[WFClient] section not found in WFCLIENT.INI”
1059	CLIENT_ERROR_HOTKEY_SHIFTSTATES_NOT_FOUND	2.1	“[Hotkey Shift States] section not found in MODULE.INI”
1060	CLIENT_ERROR_HOTKEY_KEYS_NOT_FOUND	2.1	“[Hotkey Keys] section not found in MODULE.INI”
1061	CLIENT_ERROR_KEYBOARDLAYOUT_NOT_FOUND	2.1	“[KeyboardLayout] section not found in MODULE.INI”
1062	CLIENT_ERROR_UI_ENGINE_VER_MISMATCH	2.1	“Invalid client window process”
1063	CLIENT_ERROR_IPC_TIMEOUT	2.1	“The client window process is not responding.”
1064	CLIENT_ERROR_UNENCRYPT_RECEIVED	2.1	“Data error, received unexpected unencrypted data.”
1065	CLIENT_ERROR_NENUM_NOT_DEFINED	2.1	“No network enumerator is defined in MODULE.INI”
1066	CLIENT_ERROR_NENUM_NO_SERVERS_FOUND	2.1	“No Citrix servers could be found.”
1067	CLIENT_ERROR_NENUM_NETWORK_ERROR	2.1	“A network error was encountered while searching for Citrix servers.”

Error Number	Defined Name	Version	Error String
1068	CLIENT_ERROR_NENUM_WINDOWS95_NOT_ SUPPORTED	2.1	"This feature is not supported in Windows 95. Use the Citrix server's IP address for TCP/IP or the network:node address for IPX/SPX."
1069	CLIENT_ERROR_CONNECTION_TIMEOUT	2.1	"Connection dropped because of communication errors."
1070	CLIENT_ERROR_DRIVER_UNSUPPORTED	2.1	"Driver unsupported"
1071	CLIENT_ERROR_NO_MASTER_BROWSER	2.1	"Unable to contact the Citrix Server Locator. Either your network is not functional, or you need to check your Server Location settings."
1072	CLIENT_ERROR_TRANSPORT_NOT_AVAILABLE	2.1	"The specified Network Protocol is not available."
1073	CLIENT_ERROR_NO_NAME_RESOLVER	2.1	"The specified Network Protocol is not available."
1074	CLIENT_ERROR_SEVEN_BIT_DATA_PATH	2.1	"A seven-bit data path was detected to the Citrix server. An eight-bit data path is required."
1075	CLIENT_ERROR_WIN16_WFENGINE_ALREADY_ RUNNING	2.1	"Your Citrix ICA Client connection is already running."
1076	CLIENT_ERROR_HOST_NOT_SECURED	2.1	"Cannot connect to Citrix server. The specified server is not secure."
1077	CLIENT_ERROR_ENCRYPT_UNSUPPORT_ BYCLIENT	2.1	"Your Citrix server requires encryption that your client does not support."
1078	CLIENT_ERROR_ENCRYPT_UNSUPPORT_BYHOST	2.1	"Your Citrix server does not support the encryption you required."
1079	CLIENT_ERROR_ENCRYPT_LEVEL_ INCORRECTUSE	2.1	"This connection entry uses RC5 encryption. For enhanced security, please do not specify a User name and Password in the connection properties or on the command line."
1080	CLIENT_ERROR_BAD_OVERRIDES	2.1	"(Internal error - bad connection overrides)"
1081	CLIENT_ERROR_MISSING_CONNECTION_SECTION	2.1	"(Internal error - missing connection section)"

Error Number	Defined Name	Version	Error String
1082	CLIENT_ERROR_BAD_COMBINE_ENTRIES	2.1	"(Internal error - CombinePrivateProfile Entries failed)"
1083	CLIENT_ERROR_MISSING_WFCLIENT_SECTION	2.1	"(Internal error - missing [WFClient] section)"
1084	CLIENT_ERROR_BAD_ENTRY_INSERTION	2.1	"(Internal error - AddEntrySection failed)"
1085	CLIENT_ERROR_BAD_HEADER_INSERTION	2.1	"(Internal error - AddHeaderSection failed)"
1086	CLIENT_ERROR_BAD_CONCAT_SECTIONS	2.1	"(Internal error - ConcatSections failed)"
1087	CLIENT_ERROR_MISSING_SECTION	2.1	"(Internal error - missing section)"
1088	CLIENT_ERROR_DUPLICATE_SECTIONS	2.1	"(Internal error - duplicate sections)"
1089	CLIENT_ERROR_DRIVER_BAD_CONFIG	2.1	"(Internal error - bad driver configuration)"
1090	CLIENT_ERROR_MISMATCHED_ENCRYPTION	2.1	"(Internal error - mismatched encryption)"
1091	CLIENT_ERROR_TAPI_NO_INIT	2.1	"Failed to initialize the TAPI subsystem."
1092	CLIENT_ERROR_TAPI_VER	2.1	"The required TAPI version (1.4) is not available."
1093	CLIENT_ERROR_TAPI_NO_LINES	2.1	"There are no TAPI lines defined for use."
1094	CLIENT_ERROR_TAPI_LINE_NO_EXIST	2.1	"The TAPI device specified for this dial-in connection is not defined."
1095	CLIENT_ERROR_TAPI_NEGOTIATE_LINE	2.1	"The required TAPI line cannot be opened."
1096	CLIENT_ERROR_TAPI_CALL_NOT_MADE	2.1	"The TAPI call could not be made."
1097	CLIENT_ERROR_NO_MEMORY_LOAD_AUDIO_ VESA	2.1	"No memory load audio (VESA)"
1098	CLIENT_ERROR_NO_MEMORY_LOAD_AUDIO	2.1	"No memory load audio"
1099	CLIENT_ERROR_NO_MEMORY_LOAD_VESA	2.1	"No memory load VESA"
1100	CLIENT_ERROR_NO_MEMORY_LOAD	2.1	"No memory load"

Error Number	Defined Name	Version	Error String
1101	CLIENT_ERROR_IPX_CONNECT_TIMEOUT	2.1	“The Citrix server has no more connections available at this time.”
1102	CLIENT_ERROR_NETWORK_ENUM_FAIL	2.1	“Work enumeration failed”
1103	CLIENT_ERROR_PARTIAL_DISCONNECT	2.1	“Partial disconnect”
1104	CLIENT_ERROR_CRITICAL_SECTION_IN_USE	2.1	“Critical section in use”
1105	CLIENT_ERROR_DISK_FULL	2.1	“There is insufficient space in the client directory to launch another ICA session. Contact your system administrator for assistance.”
1106	CLIENT_ERROR_DISK_WRITE_PROTECTED	2.1	“You need write privileges to the client directory to launch another ICA session. Contact your system administrator for assistance.”
1107	CLIENT_ERROR_DLL_NOT_FOUND	2.1	“A required DLL file was not found.”
1108	CLIENT_ERROR_HOST_REQUEST_FAILED	2.1	“The request to the Citrix server has failed”
1200	CLIENT_ERROR_SSL_MISSING_CACERTIFICATE	2.1	“One of the specified CACerts is missing.”
1201	CLIENT_ERROR_SSL_BAD_CACERTIFICATE	2.1	“One of the specified CACerts is corrupt.”
1202	CLIENT_ERROR_SSL_BAD_CIPHER	2.1	“One of the specified ciphers is unsupported.”
1203	CLIENT_ERROR_SSL_BAD_PRIVKEY	2.1	“The client’s private key is corrupt.”
1204	CLIENT_ERROR_SSL_UNSET_KEYSTORE	2.1	“The keystore location has not been set.”
2100	CLIENT_ERROR_SEAMLESS_HOST_AGENT_NOT_ READY	2.1	“Seamless Host agent is not ready”
2101	CLIENT_ERROR_SEAMLESS_HOST_BUSY	2.1	“Seamless Host agent is busy”
2102	CLIENT_ERROR_NOT_LOGGED_IN	2.1	“Not currently logged in”
2103	CLIENT_ERROR_SEAMLESS_CLIENT_BUSY	2.1	“Seamless Client agent is busy”
2104	CLIENT_ERROR_SEAMLESS_NOT_COMPATIBLE	2.1	“Seamless Host and Client agents are not compatible”

End User License Agreement

This is a legal agreement ("AGREEMENT") between you, the Licensed User, and Citrix Systems, Inc. or Citrix Systems International GmbH. Your location of receipt of this SDK determines the licensing entity hereunder (the applicable entity is hereinafter referred to as "CITRIX"). Citrix Systems, Inc., a Delaware corporation, markets and supports this Software Development Kit (hereinafter "SDK") in the Americas. Citrix Systems International GmbH, a Swiss company wholly owned by Citrix Systems, Inc., markets and supports this SDK in Europe, the Middle East, Africa, Asia, and the Pacific. BY USING THE SDK, YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT INSTALL OR USE THE SDK.

1. GRANT OF LICENSE. The SDK includes certain example source code, tools, utilities, program interfaces and text files to a CITRIX software product. CITRIX grants to the licensed user the following limited, non-exclusive and non-transferable rights to the SDK:

    A. Installation and Transfer. The Licensed User may install one (1) copy of the SDK on a single computer owned by the Licensed User. The Licensed User may transfer the SDK to any other computer of the Licensed User, provided that it is removed from the computer from which it is transferred. The Licensed User may make one (1) copy of the SDK in machine-readable form solely for backup purposes, provided that the Licensed User reproduces all proprietary notices on the copy.

    B. Use. The Licensed User may use the SDK solely to develop applications that access or utilize the relevant CITRIX software product.

    C. Other. Notice to Users—You shall inform all users of the SDK of the terms and conditions of this AGREEMENT.

2. DESCRIPTION OF OTHER LIMITATIONS AND OBLIGATIONS. You may not remove any proprietary notices, labels, or marks on the SDK. To the extent permitted by applicable law, you agree to allow CITRIX to audit your compliance with the terms of this AGREEMENT upon prior written notice during normal business hours.

6. LIMITATION OF LIABILITY. TO THE EXTENT PERMITTED BY APPLICABLE LAW, YOU AGREE THAT NEITHER CITRIX NOR ITS AFFILIATES, LICENSORS, OR AUTHORIZED DISTRIBUTORS SHALL BE LIABLE FOR ANY LOSS OF DATA OR PRIVACY, LOSS OF INCOME, LOSS OF OPPORTUNITY OR PROFITS, COST OF RECOVERY, LOSS ARISING FROM YOUR USE OF THE SDK, OR DAMAGE ARISING FROM YOUR PARTICIPATION IN HOSTING OR USE OF THIRD PARTY SDK OR HARDWARE OR ANY OTHER SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT, OR THE USE OF THE SDK, REFERENCE MATERIALS, OR ACCOMPANYING DOCUMENTATION, OR YOUR EXPORTATION, REEXPORTATION, OR IMPORTATION OF THE SDK, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. THIS LIMITATION WILL APPLY EVEN IF CITRIX, ITS AFFILIATES, LICENSORS, OR AUTHORIZED DISTRIBUTORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. TO THE EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL THE LIABILITY OF CITRIX, ITS AFFILIATES, LICENSORS, OR AUTHORIZED DISTRIBUTORS EXCEED $100 U.S. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU. For purposes of this AGREEMENT, the term “CITRIX AFFILIATE” shall mean any legal entity fifty percent (50%) or more of the voting interests in which are owned directly or indirectly by Citrix Systems, Inc. Affiliates, licensors, and authorized distributors are intended to be third party beneficiaries of this AGREEMENT.

7. TERMINATION. This AGREEMENT is effective until terminated. You may terminate this AGREEMENT at any time by removing the SDK from your computers and destroying all copies and providing written notice to CITRIX with the serial numbers of the terminated licenses. CITRIX may terminate this AGREEMENT at any time for your breach of this AGREEMENT. Unauthorized copying of the SDK or the accompanying documentation or otherwise failing to comply with the license grant of this AGREEMENT will result in automatic termination of this AGREEMENT and will make available to CITRIX all other legal remedies. You agree and acknowledge that your material breach of this AGREEMENT shall cause CITRIX and its licensors irreparable harm for which monetary damages alone would be inadequate and that, to the extent permitted by applicable law, CITRIX and its licensors shall be entitled to injunctive or equitable relief without the need for posting a bond. Upon termination of this AGREEMENT, the license granted herein will terminate and you must

immediately destroy the SDK and accompanying documentation, and all backup copies thereof.

8. U.S. GOVERNMENT END-USERS. If you are a U.S. Government agency, in accordance with Section 12.212 of the Federal Acquisition Regulation (48 CFR 12.212 (October 1995)) and Sections 227.7202-1 and 227.7202-3 of the Defense Federal Acquisition Regulation Supplement (48 CFR 227.7202-1, 227.7202-3 (June 1995)), you hereby acknowledge that the SDK constitutes "Commercial Computer Software," and that the use, duplication, and disclosure of the SDK by the U.S. Government or any of its agencies is governed by, and is subject to, all of the terms, conditions, restrictions, and limitations set forth in this standard commercial license AGREEMENT. In the event that, for any reason, Sections 12.212, 227.7202-1, or 227.7202-3 are deemed not applicable, you hereby acknowledge that the Government's right to use, duplicate, or disclose the SDK are "Restricted Rights" as defined in 48 CFR Section 52.227-19(c)(1) and (2) (June 1987), or DFARS 252.227-7014(a)(14) (June 1995), as applicable. Manufacturer is Citrix Systems, Inc., 851 West Cypress Creek Road, Ft. Lauderdale, Florida, 33309, or Citrix Systems International GmbH, Rheinweg 9, CH-8200 Schaffhausen, Switzerland.

9. AUTHORIZED DISTRIBUTORS AND RESELLERS. CITRIX authorized distributors and resellers do not have the right to make modifications to this AGREEMENT or to make any additional representations, commitments, or warranties binding on CITRIX or its licensors.

10. CHOICE OF LAW AND VENUE. If licensor, as defined in the preamble of this AGREEMENT, is Citrix Systems, Inc., this AGREEMENT will be governed by the laws of the State of Florida without reference to conflict of laws principles and excluding the United Nations Convention on Contracts for the International Sale of Goods, and in any dispute arising out of this AGREEMENT, you consent to the exclusive personal jurisdiction and venue in the State and Federal courts within Broward County, Florida. If licensor is Citrix Systems International GmbH, this AGREEMENT will be governed by the laws of Switzerland without reference to the conflict of laws principles, and excluding the United Nations Convention on Contracts for the International Sale of Goods, and in any dispute arising out of this AGREEMENT, you consent to the exclusive personal jurisdiction and venue of the competent courts in the Canton of Zurich. If any provision of this AGREEMENT is invalid or unenforceable under applicable law, it shall be to that extent deemed omitted and the remaining provisions will continue in full force and effect. To the extent a provision is deemed omitted, the parties agree to comply with the remaining terms of this AGREEMENT in a manner consistent with the original intent of the AGREEMENT.

11. HOW TO CONTACT CITRIX. Should you have any questions concerning this AGREEMENT or want to contact CITRIX for any reason, write to CITRIX at the following address: Citrix Systems, Inc., Customer Service, 851 West Cypress Creek Road, Ft. Lauderdale, Florida 33309; or Citrix Systems International GmbH, Rheinweg 9, CH-8200 Schaffhausen, Switzerland.
12. TRADEMARKS. Citrix is a registered trademark of Citrix Systems, Inc. in the U.S. and other countries.

SDK2_F_A45491

Index

A

about passwords 218
about scaling 218
ActiveX 19
  Embedding 19
  Events 20
  Methods 20
  Properties 20
Address 205
Application 205
APPSRVINI 205
audience 11
AudioBandwidthLimit 205
AUTHPassword 205
AUTHUsername 205
AutoLogonAllowed 205

B

BackgroundColor 205
Border 205
BorderColor 205
BrowserProtocol 206
BrowserRetry 206
Browser-Specific Properties
  BrowserProtocol 215
  DoNotUseDefaultCSL 215
  LocHTTPBrowserAddress 215
  LocIPXBrowserAddress 215
  LocNetbiosBrowserAddress 216
  LocTCPBrowserAddress 216
BrowserTimeout 206

C

CacheICAFile 206
CDMAllowed 206
ClearPassword 206
client
  requirements 17
client errors 303
ClientAudio 206
ClientName 206
ClientPath 206
ClientVersion 206
COMAllowed 206
Compress 206
Connected 206
ConnectionEntry 206
ControlWindowText 206
conventions, documentation 11
CPMAllowed 206
CustomMessage 207

D

Description 207
DesiredColor 207
DesiredHRes 207
DesiredVRes 207
DisableCtrlAltDel 207
documentation
  accessing 12
  conventions 11
  online 12
  product 12
Domain 207
DoNotUseDefaultCSL 207

E

EnableSessionSharingClient 207
EnableSessionSharingHost 213
Encrypt 207
EncryptionLevelSession 207
error codes 297
Events Interface
  OnChannelDataReceived 180
  OnClick 166
  OnConnect 167

OnConnectFailed 168
OnConnecting 169
OnDisconnect 170
OnDisconnectFailed 171
OnDisconnectSessions 194
OnDisconnectSessionsFailed 195
OnICAFile 172
OnICAFileFailed 173
OnInitializing 177
OnInitialProp 178
OnLogoffFailed 179
OnLogoffSessions 196
OnLogoffSessionsFailed 197
OnLogon 166, 174
OnPublishedApp 175
OnPublishedAppFailed 176–177
OnSessionAttach 200
OnSessionDetach 201
OnSessionEventPending 199
OnSessionSwitch 198
OnWindowCloseRequest 193
OnWindowCreated 183
OnWindowDestroyed 184
OnWindowDisplayed 192
OnWindowDocked 185
OnWindowFullscreened 190
OnWindowHidden 191
OnWindowMaximized 188
OnWindowMinimized 187
OnWindowMoved 182
OnWindowRestored 189
OnWindowSized 181
OnWindowUndocked 186
execution environment
client requirements 17
server requirements 17

F

Firefox Plug-In
Embedding 21
Events 22
Methods 22
Properties 22
Summary 22
Firefox Plug-Ins 19

G

Global log off and Disconnect
sample code 282
Global log off and disconnect
error codes 38
events 37
features 34
methods 36
properties 37
sample code 38

H

Height 207
Hotkey1Char 208
Hotkey1Shift 208
Hotkey10Char 208
Hotkey10Shift 209
Hotkey2Char 208
Hotkey2Shift 208
Hotkey3Char 208
Hotkey3Shift 208
Hotkey4Char 208
Hotkey4Shift 208
Hotkey5Char 208
Hotkey5Shift 208
Hotkey6Char 208
Hotkey6Shift 208
Hotkey7Char 208
Hotkey7Shift 208
Hotkey8Char 208
Hotkey8Shift 208
Hotkey9Char 208
Hotkey9Shift 208
HTTPBrowserAddress 207

I

ICA
error codes 297
Presentation Server Clients 16
ICA Browser
Name Resolution 215
ICA Client Object
data types 49
error codes 298
functional specification 49
Properties 204
ICA Client Object error codes 298
ICA Client Object 2.3
Connection Performance Improvements 33
Global log off and disconnect 33
new error codes 302
Support for Security Zones 24

ICAFile 209
ICAPortNumber 209
ICASOCKSProtocolVersion 209
ICASOCKSProxyHost 209
ICASOCKSProxyPortNumber 209
ICASOCKSRFC1929UserName 209
ICASOCKSTimeout 209
IconIndex 209
IconPath 209
InitialProgram 209
IPCLaunch 213
IPXBrowserAddress 209

K

KeyboardTime 209

L

LanaNumber 209
Launch 209
LocHTTPBrowserAddress 209
LocIPXBrowserAddress 210
LocNetbiosBrowserAddress 210
LocTCPBrowserAddress 210
LogAppend 210
LogConnect 210
LogErrors 210
LogFile 210
LogFlush 210
LogKeyboard 210
LogReceive 210
LogTransmit 210
LongCommandLine 214

M

Methods Interface
  AboutBox 59
  Connect 62
  CreateChannels 63
  DeleteWindow 64
  Disconnect 69
  DisconnectSessions 70
  GetClientIdentification 94
  GetClientNetworkName 95
  GetGlobalChannelCount 98
  GetGlobalChannelName 99
  GetGlobalChannelNumber 100
  GetInterfaceVersion 93
  GetLastClientError 101
  GetLastError 102
  GetMaxChannelCount 103
  GetMaxChannelRead 105
  GetMaxChannelWrite 104
  GetNotificationReason 106
  GetScreenColorDepth 108
  GetScreenHeight 114
  GetScreenWidth 116
  GetSessionColorDepth 117
  GetSessionCount 109
  GetSessionCounter 110
  GetSessionGroupCount 111
  GetSessionHandle 112
  GetSessionHandleByIndex 113
  GetSessionHeight 118
  GetSessionString 115
  GetSessionWidth 119
  GetWindowHeight 120
  GetWindowWidth 121
  GetWindowXPosition 122
  GetWindowYPosition 123
  HideTitleBar 125
  HideWindow 124
  IsConnected 126
  IsKeyboardInputEnabled 127
  IsMouseInputEnabled 128–129
  IsSessionAttached 130
  IsSessionDetached 131
  IsSessionRunning 132
  IsWindowDocked 133
  LoadIcaFile 134
  Logoff 134–135
  LogoffSessions 136
  MaximizeWindow 137
  MinimizeWindow 138
  NewWindow 139
  PlaceWindowOnBottom 140
  PlaceWindowOnTop 141
  RestoreWindow 142
  RunPublishedApplication 143
  ScaleDialog 144
  ScaleDisable 145
  ScaleDown 146
  ScaleEnable 147
  ScalePercent 148
  ScaleSize 149–151
  ScaleToFit 150
  ScaleUp 151
  SendChannelData 153
  SetChannelFlags 154
  SetSessionEndAction 155
  SetSessionGroupId 152

SetSessionId 156
SetWindowPosition 157
SetWindowSize 158
ShowTitleBar 159
Startup 160
SwitchSession 162
UndockWindow 163
MODULEINI 210
MouseTimer 210

N

Name Enumeration
when to use it 39
name enumeration 38
NetbiosBrowserAddress 210
NotificationReason 210

O

OutputMode 214

P

passwords 211
encrypted and unencrypted 218
PasswordType 211
PersistentCacheEnabled 211
Presentation Server Client error codes 303
Properties 20
Property Interface 50
ClearProps 50
DeleteProp 51
DeletePropByIndex 52
GetPropCount 53
GetPropNameByIndex 54
GetPropValue 59
GetPropValueByIndex 56
ResetProps 55, 57
SetProp 58
ProtocolSupport 211

R

Reliable 211
Requirements 11

S

scaling 218
ScalingHeight 211
ScalingMode 211
initial states 219
ScalingMode states 218
ScalingPercent 211
ScalingWidth 211
scripting 19
Scrollbars 211
server requirements 17
service and support 13
Session 214
session caching
client settings 45
methods 44
properties 44
terminology 39
when to use it 45
SessionEndAction 212
SessionExitTimeout 214
SessionGroupId 214
sessions
see ICA sessions
SessionSharingLaunchOnly 212
SessionSharingName 212
SSLCACert 212
SSLCiphers 212
SSLCommonName 212
SSLEnable 212
SSLNoCACerts 212
SSLProxyHost 212
Start 212
support and service 13

T

TCPBrowserAddress 212
TextColor 212
Title 212
TransportDriver 212
TransportReconnectDelay 212
TransportReconnectEnabled 212
TransportReconnectRetries 213
TWIDisableSessionSharing 214
TWIMode 213

U

UIActive 213
UpdatesAllowed 213
UseAlternateAddress 213
UserName 213

V

Version 213
VirtualChannels 213
VSLAllowed 213

W

WFCLIENTINI 213
Width 213
WinstationDriver 213
WorkDirectory 213

X

XmlAddressResolutionType 213