

Citrix®ICA® Client Object

API Specification Version 2.4

Programmer's Guide

ICA®/MetaFrame® Presentation Server Client for 32-bit Windows, Version 8.x

Citrix® MetaFrame® Presentation Server 3.0 for Windows®

Citrix® MetaFrame® Access Suite

Copyright and Trademark Notice

Use of the product documented in this guide is subject to your prior acceptance of the End User License Agreement. A copy of the End User License Agreement is included in Appendix 1 of this document.

Information in this document is subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Citrix Systems, Inc.

Copyright © 2001–2004 Citrix Systems, Inc. All rights reserved.

Citrix, ICA (Independent Computing Architecture), MetaFrame, NFuse, and Program Neighborhood are registered trademarks, and MetaFrame XP and SpeedScreen are trademarks of Citrix Systems, Inc. in the United States and other countries.

RSA Encryption © 1996–1997 RSA Security Inc., All Rights Reserved.

Trademark Acknowledgements

Adobe, Acrobat, and PostScript are trademarks or registered trademarks of Adobe Systems Incorporated in the U.S. and/or other countries.

Microsoft, MS-DOS, Windows, Windows Media, Windows Server, Windows NT, Win32, Outlook, ActiveX, Active Directory, and DirectShow are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Netscape, Netscape Navigator, and LiveConnect are either registered trademarks or trademarks of Netscape Communications Corporation in the U.S. and other countries.

JavaScript is a trademark of Sun Microsystems, Inc.

All other trademarks and registered trademarks are the property of their owners.

Document Code: March 25, 2004 1:39 pm (MS)

Contents

Chapter 1 Welcome	9
Getting More Information and Help	9
Disclaimer	12
Chapter 2 Introduction to the ICA Client Object	13
What Is ICA?	13
What Are ICA Clients?	13
What Is the ICA Client Object?	14
ICA Client Object Requirements	14
ICA Client Object Environment	15
Embedding and Scripting Interfaces	16
ActiveX	16
Netscape Plug-ins	18
What's New in ICA Client Object, Version 2.4	19
Security Changes	19
New Method Interface: IsPassThrough()	19
Chapter 3 ICA Client Object Features	21
Features Introduced in ICA Client Object, Version 2.3	21
ICO Connection Performance Improvements	21
ICO Global Logoff and Disconnect	21
ActiveX Control Support for Internet Explorer Security Zones	26
Features Introduced in ICA Client Object, Version 2.2	28
Session Caching	28
Name Enumeration	34
Virtual Channel Support	35
Automatic Application Resizing	38
Client Network Name and Address	42
Enable/Disable Keyboard/Mouse	43
Error Message Text From Error Code	43

Chapter 4 ICA Client Object APIs	45
Property Interface	46
ClearProps	47
DeletePropByIndex	49
GetPropCount	50
GetPropNameByIndex	51
GetPropValue	52
GetPropValueByIndex	53
ResetProps	54
SetProp	55
Methods Interface	56
AboutBox	56
AttachSession	57
CloseEnumHandle	58
Connect	59
CreateChannels	60
DeleteWindow	61
DetachSession	62
DisableKeyboardInput	63
DisableMouseInput	64
DisableSizingBorder	65
Disconnect	66
DisconnectSessions	66
DisplayWindow	68
DockWindow	69
EnableKeyboardInput	70
EnableMouseInput	71
EnableSizingBorder	72
EnumerateApplications	73
EnumerateFarms	74
EnumerateServers	75
FocusWindow	76
FullScreenWindow	77
GetCachedSessionCount	78
GetChannelData	79
GetChannelDataSize	80
GetChannelDataType	81
GetChannelCount	82
GetChannelFlags	83
GetChannelName	84
GetChannelNumber	85
GetClientAddress	86
GetClientAddressCount	87
GetErrorMessage	88

GetClientErrorMessage	89
GetInterfaceVersion	90
GetClientIdentification	91
GetClientNetworkName	92
GetEnumNameCount	93
GetEnumNameByIndex	94
GetGlobalChannelCount	95
GetGlobalChannelName	96
GetGlobalChannelNumber	97
GetLastClientError	98
GetLastError	99
GetMaxChannelCount	100
GetMaxChannelWrite	101
GetMaxChannelRead	102
GetNotificationReason	103
GetScreenColorDepth	105
GetSessionCount	106
GetSessionCounter	107
GetSessionGroupCount	108
GetSessionHandle	109
GetSessionHandleByIndex	110
GetScreenHeight	111
GetSessionString	112
GetScreenWidth	113
GetSessionColorDepth	114
GetSessionHeight	115
GetSessionWidth	116
GetWindowHeight	117
GetWindowWidth	118
GetWindowXPosition	119
GetWindowYPosition	120
HideWindow	121
HideTitleBar	122
IsConnected	123
IsKeyboardInputEnabled	124
IsMouseInputEnabled	125
IsPassThrough	126
IsSessionAttached	127
IsSessionDetached	128
IsSessionRunning	129
IsWindowDocked	130
LoadIcaFile	131
Logoff	132
LogoffSessions	133
MinimizeWindow	134

MaximizeWindow	135
NewWindow	136
PlaceWindowOnBottom	137
PlaceWindowOnTop	138
RestoreWindow	139
RunPublishedApplication	140
ScaleDialog	141
ScaleDisable	142
ScaleDown	143
ScaleEnable	144
ScalePercent	145
ScaleSize	146
ScaleToFit	147
ScaleUp	148
SetSessionGroupId	149
SendChannelData	150
SetChannelFlags	151
SetSessionEndAction	152
SetSessionId	153
SetWindowPosition	154
SetWindowSize	155
ShowTitleBar	156
Startup	157
SwitchSession	158
UndockWindow	159
Events Interface	160
OnClick	162
OnConnect	163
OnConnectFailed	164
OnConnecting	165
OnDisconnect	166
OnDisconnectFailed	167
OnICAFile	168
OnICAFileFailed	169
OnLogon	170
OnPublishedApp	171
OnPublishedAppFailed	172
OnInitializing	173
OnInitialProp	174
OnLogoffFailed	175
OnChannelDataReceived	176
OnWindowSized	177
OnWindowMoved	178
OnWindowCreated	179

OnWindowDestroyed	180
OnWindowDocked	181
OnWindowUndocked	182
OnWindowMinimized	183
OnWindowMaximized	184
OnWindowRestored	185
OnWindowFullscreened	186
OnWindowHidden	187
OnWindowDisplayed	188
OnWindowCloseRequest	189
OnDisconnectSessions	190
OnDisconnectSessionsFailed	191
OnLogoffSessions	192
OnLogoffSessionsFailed	193
OnSessionSwitch	194
OnSessionEventPending	195
OnSessionAttach	196
OnSessionDetach	197
Chapter 5 ICA Client Object Properties	199
ICA Client Object Properties	200
ICA Browser Name Resolution	208
ICA Browser-Specific Properties	209
ICA Browser Address Grid	210
About Passwords	211
About Scaling	211
Chapter 6 Example Scripts	213
Embedding	214
ActiveX	214
Netscape Plug-in	215
Scripting	216
Events	216
ActiveX	217
Netscape Plug-in	219
Logon Methods	220
ActiveX	220
Netscape Plug-in	221
Custom User Interfaces	222
ActiveX	222
Netscape Plug-in	223

Mouse Click	224
ActiveX.	224
Netscape Plug-in.	226
Scaling Methods	228
ScaleEnable and ScaleDisable	228
ScaleUp and ScaleDown	230
ScalePercent	232
ScaleToFit.	234
ScaleDialog.	236
Scaling Properties.	238
Example 1	238
Example 2.	239
Example 3.	240
Example 4.	241
Other Examples	242
ActiveX.	242
Netscape Plug-In	245
Session Caching	248
Name Enumeration.	253
Virtual Channel Support.	256
Automatic Application Resizing	262
Client Network Name and Address	270
Error Message Text from Error Codes	272
Global Logoff and Disconnect.	274
Chapter 7 Error Codes.	289
ICA Client Object Error Codes	289
Client Error Codes	295
Appendix A License Agreement	303
Index	309

Welcome

Welcome to Version 2.4 of the ICA Client Object (ICO) Specification for the ICA (Independent Computing Architecture) Client for 32-bit Windows operating systems.

This guide is for MetaFrame administrators, Web masters, ISVs, and power users of the ICA Client who are looking to integrate the ICA Client with environments and custom applications that support object embedding.

It is assumed that you have the ICA Client for 32-bit Windows (Version 8.x or later) installed and working on the client device, and is familiar with basic embedding and scripting tools and techniques.

Getting More Information and Help

This section describes how to get more information about MetaFrame Presentation Server and how to contact Citrix.

Accessing Product Documentation

The documentation for MetaFrame Presentation Server includes online documentation, known issues information, integrated on-screen assistance, and application help.

- Online documentation is provided in Adobe Portable Document Format (PDF) files. Online guides are provided that correspond to different features of MetaFrame Presentation Server. For example, information about the Web Interface is contained in the *Web Interface Administrator's Guide*. Use the *Document Center* to access the complete set of online guides.
- In many places in the MetaFrame Presentation Server user interface, integrated on-screen assistance is available to help you complete tasks. For example, in the Access Suite Console, you can position your mouse over a setting to display help text that explains how to use that control.
- Online help is available in many components. You can access the online help from the Help menu or Help button.

Important To view, search, and print the PDF documentation, you need to have Adobe Reader 5.0.5 with Search or a later version with Search. You can download Adobe Reader for free from the Adobe Web site at <http://www.adobe.com/>.

Document Conventions

MetaFrame Presentation Server 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 and option buttons, menu and tab names, and user input.
<i>Italics</i>	Placeholders for information or parameters that you provide. For example, <i>filename</i> in a procedure means you type the actual name of a file. Italics also are used for new terms and the titles of books.
UPPERCASE	Keyboard keys, such as CTRL for the Control key and F2 for the function key that is labeled F2.
Monospace	Text displayed in a text file.
%SystemRoot%	The Windows system directory, which can be WTSRV, WINNT, WINDOWS, or other name specified when Windows is installed.
{ 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 <i>devicenames</i> separated by commas.

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 that include the following:

- The Citrix Knowledge Center, an interactive tool containing thousands of technical solutions to support your Citrix environment
- Support Forums, where you can participate in technical discussions and search for previous responses from other forum members
- Access to the latest service packs, hotfixes, and utilities
- Downloadable clients, available at <http://www.citrix.com/download/>

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 will 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 MetaFrame Presentation Server

The Citrix Developer Network (CDN) is at <http://www.citrix.com/cdn/>. This open-enrollment membership program provides access to developer toolkits, technical information, and test programs for software and hardware vendors, system integrators, ICA licensees, and corporate IT developers who incorporate Citrix computing solutions into their products.

Most of the operations that you can perform using the MetaFrame Presentation Server user interface can also be scripted by using the Citrix Software Development Kit (SDK). The SDK also lets programmers customize most aspects of MetaFrame Presentation Server. The Citrix Server Software Development Kit (SDK) is available from <http://www.citrix.com/cdn/>.

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/>.

Disclaimer

This documentation is provided to you, the licensed user, by Citrix Systems, Inc. ("Citrix") pursuant to the terms of the Software Development Kit License Agreement included as Appendix A and may only be used in accordance with the express terms of the Software Development Kit License Agreement.

This document is provided "AS IS" and Citrix and its suppliers make and the licensed user receives no warranties or conditions, express, implied, statutory, or otherwise. Citrix has made reasonable efforts to ensure the completeness and accuracy of all information contained in this document.

This document is subject to change without notice and Citrix is not obligated to maintain, update, or otherwise support such documentation.

Introduction to the ICA Client Object

This chapter gives you an overview of ICA (Independent Client Architecture) and describes system requirements for the ICA Client Object and the environment in which it operates.

What Is ICA?

ICA (Independent Computing Architecture) is the foundation of Citrix server-based computing with MetaFrame Presentation Server Client software. The ICA protocol sends screen data from a computer running MetaFrame 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 ICA Clients?

An ICA Client is the software component of MetaFrame Presentation Server that executes on the client device. It allows the user to establish an ICA session with a computer running MetaFrame Presentation Server. This session enables the user to access server-based applications that appear to run locally on the client device but actually execute on the server.

ICA 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 for Workgroups, and Windows 3.x.

Web Clients (ActiveX Control and Netscape 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 (ICO) specification provides an application programming interface (API) for ICA Clients for 32-bit Windows, also known as MetaFrame Presentation Server Clients. 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, such as Internet Explorer, Netscape, and the Microsoft Office suite that support object embedding
- 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.

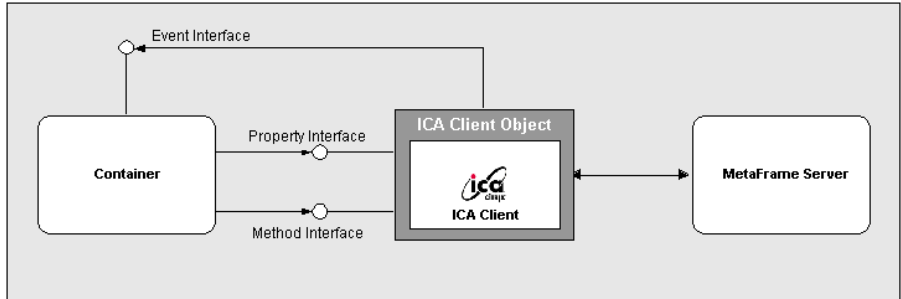
ICA Client Object Requirements

Client-Side Requirements To use ICA Client Object 2.4 functionality, you need the MetaFrame Presentation Server Client, Version 8.x or later.

Server-Side Requirements The ICA Client Object works with MetaFrame 1.8 and later.

ICA Client Object Environment

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



The ICA Client Object framework includes the following Application Programming Interfaces (APIs):

- A method interface
- A property interface
- An 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, Netscape Navigator), 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 MetaFrame Presentation Server 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, it can be controlled by 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 Netscape browser you use JavaScript.

ActiveX controls and Netscape plug-ins are two of the enabling technologies that allow you to embed objects within container applications such as Internet Explorer, Netscape, 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 MetaFrame Presentation Server 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 46 for a complete list of property interfaces available for the ICO.

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 56 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 is interested in receiving. See “[Events Interface](#)” on page 160 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.

Netscape Plug-ins

Plug-ins are a Netscape component technology. Netscape plug-ins support embedding and scripting. The Netscape plug-in API can be used to develop components that extend the capabilities of the Netscape browser.

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 Netscape 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 Netscape 3.x and above, you can set properties that are actually 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 Netscape 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 Netscape 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 46 for a complete list of property interfaces available for the ICO.

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 56 for a list of published methods.

Events

Events are intended to notify the container (Netscape Web browser) of any changes in the state of the ICA Client Object. See “[Events Interface](#)” on page 160 for a list of available events.

Summary

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

What's New in ICA Client Object, Version 2.4

Security Changes

The following events can now be used in the Internet security zone of Internet Explorer:

- OnConnect
- OnConnectFailed
- OnDisconnect
- OnIcaFile
- OnIcaFileFailed

Note Restricted sites and Internet zones can download only HTTP and HTTPS type ICA files.

For a complete list of Internet Explorer security zones and events permitted to initialize in those zones, see “ActiveX Control Support for Internet Explorer Security Zones” on page 26.

New Method Interface: IsPassThrough()

The new method interface, IsPassThrough, determines whether the container application is running within a server session or on the local client device.

Methods

Method	Description
IsPassThrough	Determines whether the container application is running within a server session or on the local client device.

ICA Client Object Features

Features Introduced in ICA Client Object, Version 2.3

This section describes new features and functionality added in Version 2.3 of the ICA Client Object specification.

ICO 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 non-blocking 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 ICO sessions are started from a Web page.

ICO Global Logoff and Disconnect

Global logoff 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
Logoff 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 logoff.
Switching between launched sessions	This allows users to switch between 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 Logoff and Disconnect events have been added.

Features

Ability to Disconnect and Logoff Multiple Sessions

Use the DisconnectSessions and LogoffSessions APIs to disconnect or logoff multiple sessions. Any ICO 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 logoff command. OnDisconnectSessions and OnLogoffSessions are fired if successful; OnDisconnectSessionsFailed and OnLogoffSessionsFailed are fired 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.

Ability to Group Sessions Using SessionGroupId

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

The default value of `SessionGroupId` is “ICOGroup” and ICO sessions for which you do not specifically set `SessionGroupId` are assigned “ICOGroup” as the session group ID. `SessionGroupId` can be set 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 logoff 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 logoff or disconnect.

Ability to Launch Sessions Using IPC

The implementation of Inter Process Communication (IPC) support for ICA sessions launched through ICO 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 logoff 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 may be required for users of Netscape, or for users who want to use the ICA Connection Center.

Ability to Switch between Launched Sessions

To enable ICO 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 can be the actively selected session inside ICO at a time. This includes launched sessions. To receive events from a different launched session, you need to switch to that session. This is also true for all commands (for example, logoff 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 ICO session is represented by its session handle. The session handle is relevant to a specific ICO object only and is matched with a launched session using an internal linked list.

Session handles are specific to an ICO object; they cannot be used to access sessions launched from other ICO objects. This is an effective security mechanism to protect sessions from malicious access. The only session handles available to a particular ICO 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 API to return zero as an error context for the session handle.

Methods, Events, Properties, and Error Codes

Methods

Method	Description
DisconnectSessions	Disconnect all sessions that have the designated SessionGroupld.
LogoffSessions	Logoff all sessions that have the designated SessionGroupld.
SetSessionGroupld	Set the session's Group ID.
GetSessionHandle	Get the handle for the current session.
SwitchSession	Set the current session handle that selects a new session.
GetSessionCount	Get the number of active started sessions.
GetSessionHandleByIndex	Get the launched session handle using an index.
GetSessionGroupCount	Get the number of sessions still running that belong to the specified group.

Events

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

Properties

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

Error Codes

The following error codes apply to the global logoff 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 SessionGroupIid.
ICO_ERROR_SESSION_ALREADY_SELECTED	Session is already selected.
ICO_ERROR_CANNOT_SCRIPT_IN_ZONE	The Citrix ICA Client Object cannot be scripted in the Internet or Restricted sites zones.

You can use ICO Global Logoff and Disconnect to group launched sessions and logoff or disconnect session groups. This feature is primarily designed to enable removal of all sessions simultaneously.

Sample Code

For an example script that uses the global disconnect or logoff methods, see [“Global Logoff and Disconnect”](#) on page 274.

ActiveX Control Support for Internet Explorer Security Zones

The MetaFrame Presentation Server ActiveX control is digitally signed by Citrix, rendering it safe for scripting and initialization.

In earlier releases, some ICO 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 ICO 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 ICO 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 ICO 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
Trusted Sites	Yes	Yes
Restricted Sites	No	No
Internet	No	Initialization is permitted for the following parameters and events: Parameters - Start, Border, Width, Height, SRC, CacheICAFFile, SessionId, SessionCacheEnable, AutoAppResize, and ICAFile. Events - OnConnect, OnConnectFailed, OnDisconnect, OnIcaFile, OnIcaFileFailed.

Note Restricted sites and Internet zones can download only HTTP and HTTPS type ICA files.

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. ActiveX controls must not be used in the Restricted Sites zone, and are disabled by default in that zone in Microsoft Internet Explorer.

To allow backward compatibility, an additional setting, IgnoreZones, is included in Webica.ini that disables the above checks. To disable the above 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 ICO control. Scripting functionality is disabled if security zones are set up incorrectly.

Features Introduced in ICA Client Object, Version 2.2

This section describes new features and functionality added in Version 2.2 of the ICA Client Object specification.

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 functionality 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 automatically displayed. The session caching feature introduces several new terms as listed below:

Term	Definition
Attached	This implies that the session instance is connected and associated with an instance of ICO.
Detached	This implies that the session instance is not associated with an instance of the ICO. The session instance is said to be <i>cached</i> .
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 ICO.
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 ICO. The session instance is said to be <i>cached</i> .

Features

Session caching features can be used to significantly improve the user experience of the client in a Web environment. Session caching has the following 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 will be 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 utilizing too much 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, if there are five sessions and the cache limit is reduced to three. Three of the five sessions need to 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

Provides the ability 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 ICO to a session. Without a SessionId, it is difficult to ascertain 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 ICOs. You can use non-unique session identifiers if you are not particular about placing a session within the right ICO container. For example, if you have two detached sessions running the same application, and it doesn't 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

Implicit mode provides the ability 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 ICO. 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 ICO if you are using Implicit mode. To use ICO in Implicit mode, enable session caching and specify a valid SessionId. You must also set a non-zero time-out and a suitable cache limit value.

Explicit Mode

Explicit mode provides the ability 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 “attach,” “detach,” “set SessionId,” “query if session is cached,” and “get a count of cached sessions” are available. Note that Explicit mode is not affected by the Cache Enable flag.

Mixed Mode

Mixed mode provides the ability 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 ICO startup and shutdown functions. For example, if a “detach” is performed before ICO shutdown, the Implicit mode is informed about the detach operation and does not attempt to detach again. If Implicit mode attaches a session during ICO startup, an explicit call is allowed any time afterwards to attach a different cached session. The “attach” API automatically detaches the existing session.

ICO Startup

ICO startup provides support for automatic reattaching of cached sessions for Implicit mode.

When you create an instance of the ICO, 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, ICO reattaches the detached session. If the attach works, ICO ignores the setting for the Start property. This prevents problems such as starting a duplicate session. ICO reestablishes ICO state information when the session is reattached.

ICO Shutdown

ICO shutdown provides support for automatic detaching of sessions for Implicit mode.

When an instance of ICO 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 ICO is detached. If the limit of cached sessions is exceeded, the cached session that is most deserving is detached. ICO 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 ICO.

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 ICO container that created it. This protects against malicious programs/scripts that may try to reestablish sessions that are already logged on to a server. Certain environments may 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

Methods

Name	Description
AttachSession	Attach to a cached ICA session.
DetachSession	Detach from an ICA session.
GetCachedSessionCount	Get the count of cached sessions.
IsSessionAttached	Determine if session is currently attached.
IsSessionDetached	Determine if session is currently cached.
IsSessionRunning	Determine if session is currently running.
SetSessionId	Set the session ID for the current session.

Properties

Name	Description
SessionCacheEnable	Enable/disable the session caching feature.
SessionCacheTimeout	Length of time in seconds to keep cached session.
SessionId	Unique identifier for 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	Enable or disable the session caching feature. This setting cannot be overridden (master switch to turn session caching on or off).
SessionCacheEnable	Enable or disable the session caching feature (default setting for client sessions).
SessionCacheTimeout	Length of time in seconds to keep 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 248.

When to Use Session Caching

Session caching functionality is designed to provide 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 between multiple ICA sessions within a single ICO 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 ICO 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.

Name Enumeration

Prior to this release, it was not possible to enumerate servers or applications from an instance of the ICA Client Object. To connect to a server or a published application, you needed to hard-code the name of the resource.

ICO Version 2.2 and later provides name enumeration functionality that enables you to select a server or application from the enumeration list. You can use the name enumeration functionality in a script to access names of servers, applications, and farms. To get an enumeration list, the name enumeration function needs to be passed a valid browser address (TCPBrowserAddress, HTTPBrowserAddress, and so on).

Name Enumeration Methods

Name	Description
EnumerateServers	Enumerate a list of available servers.
EnumerateApplications	Enumerate a list of available applications.
EnumerateFarms	Enumerate a list of available server farms.
GetEnumNameCount	Get the number of enumerated names.
GetEnumNameByIndex	Get the enumerated name by index number.
CloseEnumHandle	Close the enumeration handle and free memory.

For descriptions of the above methods, see “[ICA Client Object APIs](#)” on page 45. For code examples incorporating name enumeration functionality, see “[Name Enumeration](#)” on page 253.

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. Name enumeration makes it possible to use scripting to dynamically refresh enumeration lists for published resources and notify you of updates.

Virtual Channel Support

Prior to this release, the only way you could access virtual channels was to develop a virtual driver for the channel.

With the introduction of virtual channel support in ICO 2.2, you can 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 BSTR support. BSTRs are not null terminated and are specified in terms of their lengths. This means that BSTRs can contain binary data that do 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

Methods

Name	Description
CreateChannels	Create virtual channels based on the list passed in.
GetChannelCount	Get the number of channels created from CreateChannels.
GetChannelDataSize	Get the size of the current incoming data.
GetChannelDataType	Get the type of data for the incoming data (string or binary).
GetChannelData	Get the data for the incoming request.
GetChannelFlags	Get the flags for the virtual channel.
GetChannelName	Get the name for the virtual channel by index (maximum from GetChannelCount minus one).
GetChannelNumber	Get the virtual channel number for the virtual channel name.
GetGlobalChannelCount	Get the total number of virtual channels for the client.
GetGlobalChannelName	Get the name of the virtual channel by the index.
GetGlobalChannelNumber	Get the virtual channel number of the virtual channel by name.
GetMaxChannelCount	Get the maximum number of virtual channels supported.
GetMaxChannelRead	Get the maximum number of bytes that can be read at once.
GetMaxChannelWrite	Get the maximum number of bytes that can be written at once.
SendChannelData	Set data over the virtual channel.
SetChannelFlags	Set the flags for the virtual channel.

Properties

Name	Description
VirtualChannels	List of virtual channel names to create.

Events

Name	Description
OnChannelDataReceived	Notification that virtual channel data is available.

Usage

Virtual channels are primarily used 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 ICO now enables you to use virtual channels to create new resource types or to exchange information between client and server. A good example of this is a custom application on the client communicating with a custom application on the server. Using ICO virtual channel support, you can allow exchange of any type of data without being concerned about the actual connection.

Virtual channel support in ICO 2.2 is extremely 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 has 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 256.

Automatic Application Resizing

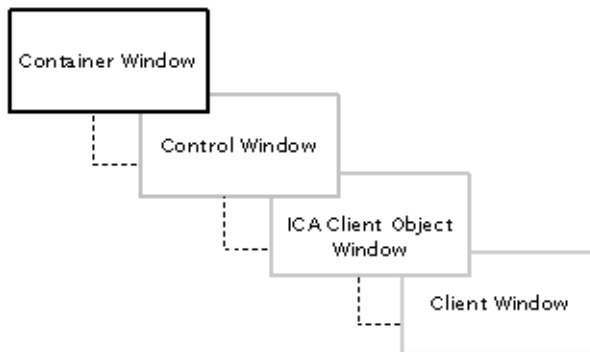
This functionality allows you to resize a server application window within the ICA Client Object container. In the past, when the ICO container was resized, the server application window size remained intact. With automatic application resizing, you can now resize the server application window to the 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 ICO 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 ICO 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 ICO windows. The Z ordering hierarchy of ICO windows is quite strict; that is, the container window is the parent of the control window; which is the parent of the ICO window, which is the parent of the client window.



Use the `AutoAppResize` property available in ICO 2.2 to enable automatic application resizing. The default value of `AutoAppResize` is `False`.

ICO Window Types

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

ICO Window Position Flags

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

ICO Window Area Flags

Area Type	Identifier	Value
Client area	ICO_WINDOW_FLAG_SIZE_CLIENT_AREA	0
Window area	ICO_WINDOW_FLAG_SIZE_WINDOW_AREA	1

ICO New Window Flags

New Window Flag	Identifier	Value
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

Name	Description
GetWidthWidth	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 ICO window on top of other windows.
PlaceWindowOnBottom	Place the ICO window at the very bottom of other windows.
ShowTitleBar	Show the title bar on the undocked ICO window.
HideTitleBar	Hide the title bar on the undocked ICO window.
EnableSizingBorder	Enable the sizing border on the undocked ICO window.
DisableSizingBorder	Disable the sizing border on the undocked ICO window.
FocusWindow	Give the ICO window the keyboard focus.
UndockWindow	Undock the ICO window from the control window.
DockWindow	Dock the ICO window in the control window.
MinimizeWindow	Minimize the ICO window.
MaximizeWindow	Maximize the ICO window.
RestoreWindow	Restore the ICO window to the original size.
FullScreenWindow	Make the ICO window a full screen window.
IsWindowDocked	Determine if a window is currently docked.
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.

Name	Description
GetScreenColorDepth	Get the color depth for the screen. The number returned is bits per pixel for color.
NewWindow	Create a new ICO window (only if the ICO window no longer exists).
DeleteWindow	Delete the ICO window.

Properties

Name	Description
AutoScale	Automatically scale the client session.
AutoAppResize	Automatically resize the client application to fit the ICO window.

Events

Event #	Handler	Description
16	OnWindowSized	The window size is changed.
17	OnWindowMoved	The window position is changed.
18	OnWindowCreated	The window is created.
19	OnWindowDestroyed	The window is destroyed.
20	OnWindowDocked	The window is docked.
21	OnWindowUndocked	The window is undocked.
22	OnWindowMinimized	The window is minimized.
23	OnWindowMaximized	The window is maximized.
24	OnWindowRestored	The window is restored.
25	OnWindowFullscreened	The window is set to full screen.
26	OnWindowHidden	The window is hidden.
27	OnWindowDisplayed	The window is displayed.
28	OnWindowCloseRequest	The window is being requested to close.

For descriptions of the above methods, properties, and events, see “[ICA Client Object APIs](#)” on page 45. For code examples incorporating automatic application resizing functionality, see “[Automatic Application Resizing](#)” on page 262.

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 as desired through scripting. The interfaces provided with this feature enable you to do powerful, dynamic, and transparent operations with ICO windows.

Client Network Name and Address

This feature allows the network name and address to be retrieved from ICO. This is one of the most wanted features in ICO because it provides the ability 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 DHCP or locally configured settings.

Methods

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

For descriptions of the above methods, see “[ICA Client Object APIs](#)” on page 45. For code examples incorporating these methods, see “[Client Network Name and Address](#)” on page 270.

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.

Enable/Disable Keyboard/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

Name	Description
DisableKeyboardInput	Disable keyboard input for the session.
DisableMouseInput	Disable mouse input for the session.
EnableKeyboardInput	Enable keyboard input for the session.
EnableMouseInput	Enable 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 45. For code examples incorporating these methods, see “[Automatic Application Resizing](#)” on page 262.

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

The ICO 2.2 specification makes it possible to extract error message text for a particular error code. This was not possible in previous versions unless the application or container had its own error table for ICO and the client.

Function

The function is fairly simple. Given an error code from ICO or the client, these methods return the corresponding error message text. The error messages are built into ICO and are used for normal error processing; however these messages can also be used by the container when required.

Methods

Name	Description
GetErrorMessage	Get the ICO error message for error number.
GetClientErrorMessage	Get the client error message for error code.

For descriptions of the above methods, see [“ICA Client Object APIs”](#) on page 45. For code examples incorporating these methods, see [“Error Message Text from Error Codes”](#) on page 272.

Usage

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

ICA Client Object APIs

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

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 may differ.

Data Type or Value	Remarks
BOOLEAN	Boolean value, only TRUE or FALSE
STRING	An 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 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 is started, 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 will 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 will 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 may 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

Important DeletePropByIndex will 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 may 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, zero length string otherwise

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.

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

Calling convention

STRING GetPropValueByIndex(NUMBER *Index*)

Return value

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

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 will not reset read-only properties. Current values for read-only properties will be 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 will 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, containing copyright and version information about the ICA Client Object, on the client device.

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

Close 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 `GetLastError`.

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. Launch status information is returned by the `OnConnect` or `OnConnectFailed` events.

CreateChannels

Specifies the virtual channel names to be created for the session. Only one parameter can be specified 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 only be called 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 in conjunction 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 will detach 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 – ICO Error Code

Call time

Run-time

Supported in

ICA Client Object 2.2 or later

Remarks

DisableKeyboardInput only works 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 highly desirable. It is also useful for monitoring sessions without worrying about input from keyboard and mouse.

DisableMouseInput

Disable the mouse input for the session.

Calling convention

NUMBER DisableMouseInput(VOID)

Parameters

None

Returns

NUMBER – ICO 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 keyboard and mouse.

Important This function works only within the ICO window. The parent or container window (IE or Netscape) 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

This function 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 GetLastErrorClientError.

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 container is notified of disconnection status by the OnDisconnect or OnDisconnectFailed events.

DisconnectSessions

Disconnects a group of sessions based on 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 ICO 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 OnDisconnectSessions and OnDisconnectSessionsFailed are used to 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	ICO Window
1	Control Window
2	Client Window

Returns

NUMBER — ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

DisplayWindow is useful for displaying windows that were hidden using HideWindow.

DockWindow

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

Calling convention

NUMBER DockWindow(VOID)

Parameters

None

Returns

NUMBER – ICO 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 – ICO 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 highly desirable. These API are also useful for monitoring sessions without worrying about keyboard and mouse input.

EnableMouseInput

Enables mouse input for the session.

Calling convention

NUMBER EnableMouseInput(VOID)

Parameters

None

Returns

NUMBER – ICO 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 highly desirable. These API are also useful for monitoring sessions without worrying about keyboard and 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 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 – ICO 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 ICO window. This ensures that all keyboard input is directed to the ICO window.

FullScreenWindow

Set the current window size to full screen.

Calling convention

NUMBER FullScreenWindow(VOID)

Parameters

None

Returns

NUMBER – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

FullScreenWindow is useful to force 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 Netscape. Binary is possible only when using the ICO 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 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 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 is useful in determining 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 has only the ability to determine whether or not the write requests will 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

Use `GetChannelName` to determine 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

Use GetChannelNumber to determine 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 machine 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 allows for 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 machines have only one TCP/IP address but some have more than one. GetClientAddressCount enables you to determine 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 ICA Client, see [“Error Codes”](#) on page 289. The error message text returned will be in the language supported by the ICA Client for 32-bit Windows.

GetClientErrorMessage

Get the error message text for the ICA 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 [“Error Codes”](#) on page 289. The error message text returned will be in the language supported by the ICA Client for 32-bit Windows.

GetInterfaceVersion

This function gets the ICA Client Object API version as a string value. The API version has major and minor values separated by a period (that is, 2.0). GetInterfaceVersion determines what methods, properties, and events are supported by a specific version of ICO.

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

This function gets a string that completely identifies the ICA Client for 32-bit Windows using the client update identification data. 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, “foo” is returned by GetClientNetworkName. 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 ICA 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/disabled during load time and also to determine what is running. These names include the names we create using CreateChannels.

GetGlobalChannelNumber

Gets 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.

GetLastError

This function returns the last error code from the ICA Client.

Calling convention

NUMBER GetLastError(VOID)

Call time

Load-time, run-time

Supported in

ICA Client Object 2.1 or later

Remarks

Only valid values are returned for certain methods. All other calls generate a zero error code. For a list of ICA Client Object and Client Error Codes, see “[Error Codes](#)” on page 289.

GetLastError

This function 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 ICO method. For a list of ICA Client Object and Client Error Codes, see [“Error Codes”](#) on page 289.

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

Use GetMaxChannelCount to determine 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

Gets 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

GetMaxChannelWrite is used 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 becomes less important.

GetMaxChannelRead

Gets 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 primarily based 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 has failed
3	Logon has 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	Logoff 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 ICO 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 ICO instance.

Calling convention

NUMBER GetSessionCount(VOID)

Parameters

None

Returns

NUMBER SessionCount — sessions in an ICO 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 may 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 `GetLastError`.

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

Get 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

Use GetSessionGroupCount to determine 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 ICO instances with the same session group name. The value of GetSessionGroupCount may be greater than the value of GetSessionCount because GetSessionCount applies only to the current ICO instance.

GetSessionHandle

Gets the currently active session handle.

Calling convention

HANDLE GetSessionHandle(VOID)

Parameters

None

Returns

HANDLE Session — handle to 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

Get 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 is useful for enumerating the different handles for sessions in an ICO 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

Use GetScreenHeight to determine the DesiredHRes property. Determine the real local desktop size and then work out the DesiredHRes of the remote desktop.

GetSessionString

This function 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 `GetLastError`.

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

Use GetScreenWidth to determine 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

Use GetSessionColorDepth to determine what color depth was used to connect to the server. A session must already be 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

Use GetSessionHeight to determine 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

Use GetSessionWidth to determine 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 | ICO 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.

GetWidthWidth

Returns the width of the selected window type in pixels.

Calling convention

NUMBER GetWindowWidth(NUMBER *WndType*,
NUMBER *WndFlags*)

Parameters

NUMBER *WndType*

- | | |
|---|------------------|
| 0 | ICO 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

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

Calling convention

NUMBER GetWindowXPosition(NUMBER *WndType*,
NUMBER *WndFlags*)

Parameters

NUMBER *WndType*

- | | |
|---|------------------|
| 0 | ICO 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 – 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

Use GetWindowXPosition to determine the position of the relevant window types. Client window does not exist until a connection is started.

GetWindowYPosition

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

Calling convention

NUMBER GetWindowYPosition(NUMBER *WndType*,
NUMBER *WndFlags*)

Parameters

NUMBER *WndType*

- | | |
|---|------------------|
| 0 | ICO 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

This API is useful 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 | ICO Window |
| 1 | Control Window |
| 2 | Client Window |

Returns

NUMBER – ICO 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 ICO window.

Calling convention

NUMBER HideTitleBar(VOID)

Parameters

None

Returns

NUMBER – ICO 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 ICO 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

This function gets 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

Remarks

None

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

Use IsSessionAttached to check if the session to which you are trying to attach is already attached. You can also use IsSessionAttached to check the state of a session(s).

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 ICO

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use IsSessionDetached to determine 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

Use IsSessionRunning to determine if the session still exists regardless of its attached or detached state.

IsWindowDocked

Determines if the ICO window is currently docked.

Calling convention

BOOLEAN IsWindowDocked(VOID)

Parameters

None

Returns

TRUE if the ICO window is docked; FALSE if it's undocked

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use IsWindowDocked to determine whether or not the window is currently undocked.

LoadIcaFile

This function 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

This function initiates the asynchronous download of the ICA file from the specified location. The container is notified about ICA file download status by OnICAFile or OnICAFileFailed events.

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

This function 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 `GetLastError`.

Calling convention

`VOID Logoff(VOID)`

Call time

Run-time

Supported in

ICA Client Object 2.0 or later

Remarks

Logoff status notification is returned by the `OnDisconnect` or `OnLogoffFailed` events.

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

LogoffSessions

Log off 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 log off. All sessions in the specified group must have the same SessionGroupId to qualify for log off 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 log out to complete.

MinimizeWindow

Minimizes the undocked window.

Calling convention

NUMBER MinimizeWindow(VOID)

Parameters

None

Returns

NUMBER – ICO 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.

MaximizeWindow

Maximizes the undocked window.

Calling convention

NUMBER MaximizeWindow(VOID)

Parameters

None

Returns

NUMBER – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

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

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 – ICO 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 ICO Window underneath other windows.

Calling convention

NUMBER PlaceWindowOnBottom(VOID)

Parameters

None

Returns

NUMBER – ICO 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 ICO window above other windows from the same parent.

Calling convention

NUMBER PlaceWindowOnTop(VOID)

Parameters

None

Returns

NUMBER – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use PlaceWindowOnTop to ensure better visibility of the ICO 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 – ICO 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

This function launches a published application within a connected session. This function supports `GetLastError`.

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 that argument lengths are supported differently depending on the version of MetaFrame Presentation Server. MetaFrame 1.8, Feature Release 1, supports up to 256 bytes of argument information, MetaFrame XP and later supports 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 “%*” 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. Status of the launch is returned by `OnPublishedApp` and `OnPublishedAppFailed` events.

ScaleDialog

This function displays the scaling dialog box of the client session window. This function supports GetLastError.

Calling convention

NUMBER ScaleDialog(VOID)

All scale functions return a number. This number is the error code for ICO. Potentially more information could be obtained using GetLastError.

Returns

NUMBER – ICO 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 ICA Client for 32-bit Windows without ICO functionality.

ScaleDisable

This function disables scaling of the client session window. This function supports GetLastError.

Calling convention

NUMBER ScaleDisable(VOID)

Returns

NUMBER — ICO 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

This function shrinks the size of the client session window. This function supports GetLastError.

Calling convention

NUMBER ScaleDown(VOID)

Returns

NUMBER — ICO 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

This function enables scaling of the client session window. This function supports GetLastError.

Calling convention

NUMBER ScaleEnable(VOID)

Returns

NUMBER — ICO 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

This function scales the client session window size by a specified percentage value. This function supports GetLastError.

Calling convention

NUMBER ScalePercent(NUMBER *Percent*)

Returns

NUMBER — ICO error code

Call time

Run-time

Supported in

ICA Client Object 2.1 or later

Remarks

This function 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

This function scales the client session window to a specified size. This function supports `GetLastError`.

Calling convention

NUMBER ScaleSize(NUMBER *Width*,
NUMBER *Height*)

Returns

NUMBER — ICO error code

Call time

Run-time

Supported in

ICA Client Object 2.1 or later

Remarks

This function 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

This function scales the client session window to fit the size of the control window. This function supports GetLastError.

Calling convention

NUMBER ScaleToFit(VOID)

Returns

NUMBER — ICO 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 tells the client to set the area of the scaled image to be the same as the control window.

ScaleUp

This function enlarges the client session window. This function supports GetLastError.

Calling convention

NUMBER ScaleUp(VOID)

Returns

NUMBER – ICO error code

Call time

Run-time

Supported in

ICA Client Object 2.1 or later

Remarks

This function 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 – ICO return code

Call time

Run-time

Supported in

ICA Client Object 2.3 or later

Remarks

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 — ICO 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' would correspond 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 virtual channel

NUMBER Flag

0 Do not fragment packets

1 Fragment packets

Returns

NUMBER — ICO error code

Call time

Run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use SetChannelFlags to enable or disable packet fragmenting. Enabling packet fragmenting allows transmission of larger packets of data.

SetSessionEndAction

This function 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

Set 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 – ICO 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 that you cannot get the value for the SessionId once it is set.

SetWindowPosition

Sets the position of the ICO or control window relative to the parent.

Calling convention

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

Parameters

NUMBER *WndType*

0 ICO 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 – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use SetWindowPosition to position the ICO window when it is undocked.

SetWindowSize

Sets the size of the ICO or control window.

Calling convention

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

Parameters

NUMBER *WndType*

0 ICO 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 – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use SetWindowSize to resize the control or ICO windows. It is particularly useful to resize the ICO window when it is undocked.

ShowTitleBar

Shows the title bar for the ICO window.

Calling convention

NUMBER ShowTitleBar(VOID)

Parameters

None

Returns

NUMBER – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Usually the ICO window already has a title bar. However, ShowTitleBar is useful when the title bar is explicitly hidden.

Startup

This function 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 – ICO 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 ICO window on the desktop.

Calling convention

NUMBER UndockWindow(VOID)

Parameters

None

Returns

NUMBER – ICO error code

Call time

Load-time, run-time

Supported in

ICA Client Object 2.2 or later

Remarks

Use UndockWindow to convert 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 Netscape 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 4, "ICA Client Object Properties," for more information.

Supported in

ICA Client Object 2.0 or later

OnConnect

The ICA 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. Possible causes of failure can be determined from `GetLastError` and `GetLastClientError`.

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 logoff notification. When this notification event occurs, the ICA Client is no longer available, and ICO returns to load-time mode for the methods and properties.

Calling convention

```
VOID OnDisconnect()
```

Supported in

OnDisconnectFailed

Disconnect failure notification. This event is a result of a failed Disconnect method call.

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

Published application launched successfully notification. 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

Failure to launch published application notification. 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

Logoff failure notification. 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 ICO window. This permits some control over how the ICO window is shutdown. If nothing is done, the ICO 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 GetLastError 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 ICO. 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 ICO 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 non-selected session.

Calling convention

OnSessionEventPending (NUMBER hSession, NUMBER eventId)

Supported in

ICA Client Object 2.3 or later

Remarks

This event is used to inform 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 160](#). 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 ICO 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 ICO instance.

ICA Client Object Properties

This chapter describes the commonly used properties supported by the ICA Client Object. 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 that 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

Name	Description	Type	Ver.	Valid Value(s)
Address	Target server address (network name or physical address).	String	2.0	
Application	Published application name. If defined, it overrides the Address and InitialProgram properties.	String	2.0	
APPSRVINI	Full path and filename of appsrv.ini.	String	2.0	
AudioBandwidthLimit	Audio bandwidth limit or audio quality.	Number	2.0	0=high, 1=medium (default) 2=low
AUTHUsername	SSL authorization username.	String	2.2	
AUTHPassword	SSL authorization password.	String	2.2	
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 #FFFFFF=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) #FFFFFF=white other "#RRGGBB" values
BrowserProtocol	Protocol to use for ICA browser name resolution.	String	2.0	IPX, SPX, NetBIOS, UDP, HTTPonTCP
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)

Name	Description	Type	Ver.	Valid Value(s)
CacheICAFile	Determine whether or not the ICA file will be cached by the ICA Client Object.	Boolean	2.2	TRUE (default), FALSE
CDMAAllowed	Enable support for client drive mapping.	Boolean	2.0	TRUE (default), FALSE
ClearPassword	Clear text password for authentication.	Write-only string	2.2	
ClientName	Unique name of client.	String	2.0	Defaults to the current ICA Client setup.
ClientAudio	Enable client audio support.	Boolean	2.0	FALSE (default), TRUE
ClientPath	Path of ICA Client (read-only).	Read-Only String	2.0	
ClientVersion	Version of ICA 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	
ControlWindowText	Text to display in control when UIActive is TRUE, and window is undocked.	String	2.2	
CPMAAllowed	Allow client printer port mapping.	Boolean	2.0	TRUE (default), FALSE
CustomMessage	Custom message to display in ICA Client Object when not connected. Must also set UIActive to TRUE.	String	2.0	
Description	General description of connection.	String	2.0	
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

Name	Description	Type	Ver.	Valid Value(s)
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	
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
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	
Hotkey1Char	Hotkey 1 character (Task List).	String	2.2	"F1" (default)
Hotkey1Shift	Hotkey 1 Shift character (Task List).	String	2.2	"Shift" (default)
Hotkey2Char	Hotkey 2 character (close remote application).	String	2.2	"F3" (default)
Hotkey2Shift	Hotkey 2 Shift character (close remote application).	String	2.2	"Shift" (default)
Hotkey3Char	Hotkey 3 character (toggle title bar).	String	2.2	"F2" (default)
Hotkey3Shift	Hotkey 3 Shift character (toggle title bar).	String	2.2	"Shift" (default)
Hotkey4Char	Hotkey 4 character (Ctrl-Alt-Del).	String	2.2	"F1" (default)
Hotkey4Shift	Hotkey 4 Shift character (Ctrl-Alt-Del).	String	2.2	"Ctrl" (default)

Name	Description	Type	Ver.	Valid Value(s)
Hotkey5Char	Hotkey 5 character (Ctrl-Esc).	String	2.2	"F2" (default)
Hotkey5Shift	Hotkey 5 Shift character (Ctrl-Esc).	String	2.2	"Ctrl" (default)
Hotkey6Char	Hotkey 6 character (Alt-Esc).	String	2.2	"F2" (default)
Hotkey6Shift	Hotkey 6 Shift character (Alt-Esc).	String	2.2	"Alt" (default)
Hotkey7Char	Hotkey 7 character (Alt-Tab).	String	2.2	"plus" (default)
Hotkey7Shift	Hotkey 7 Shift character (Alt-Tab).	String	2.2	"Alt" (default)
Hotkey8Char	Hotkey 8 character (Alt-Backtab).	String	2.2	"minus" (default)
Hotkey8Shift	Hotkey 8 Shift character (Alt-Backtab).	String	2.2	"Alt" (default)
Hotkey9Char	Hotkey 9 character (Ctrl-Shift-Esc).	String	2.2	"F3" (default)
Hotkey9Shift	Hotkey 9 Shift character (Ctrl-Shift-Esc).	String	2.2	"Ctrl" (default)
Hotkey10Char	Hotkey10 character (toggle latency reduction).	String	2.2	"F4" (default)
Hotkey10Shift	Hotkey 10 Shift character (toggle SpeedScreen Latency Reduction).	String	2.2	"Ctrl" (default)
ICAFile	Name of ICA file to use for connection. URLs are allowed.	String	2.0	
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.	Number	2.2	
ICASOCKSProxyPortNumber	Port number for SOCKS support.	Number	2.2	1080 (default)
ICASOCKSRFC1929UserName	SOCKS authorization user name.	String	2.2	
ICASOCKSRFC1929Password	SOCKS authorization password.	String	2.2	
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	

Name	Description	Type	Ver.	Valid Value(s)
IconPath	Icon file pathname.	String	2.0	
InitialProgram	Initial program to run on server.	String	2.0	
IPXBrowserAddress	IPX browser address for application server-name resolution.	String	2.0	
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	
LocIPXBrowserAddress	Local IPX browser address for application/server-name resolution.	String	2.0	
LocNetbiosBrowserAddress	Client local NetBIOS browser address for application/server-name resolution.	String	2.0	
LocTCPBrowserAddress	Local TCP/IP browser address for application/server-name resolution.	String	2.0	
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	
LogFlush	Flush out log results for each write (very slow). All the log data is written out 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)

Name	Description	Type	Ver.	Valid Value(s)
LogTransmit	Log the transmit data.	Boolean	2.0	TRUE, FALSE (default)
MODULEINI	Pathname for module.ini file (optional).	String	2.0	
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	
NotificationReason	Reason for last event notification.	Read-Only Number	2.0	See event table
Password	User password for connection. Internal user based on PasswordType. All passwords are stored in encrypted form. See "About Scaling" on page 211.	Write-Only String	2.0	
PasswordType	Type of password being used. See "About Scaling" on page 211.	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	
Reliable	Enable reliable protocol support.	Boolean	2.0	TRUE (default), FALSE
ScalingHeight	Height of scaled window. See "About Scaling" on page 211.	Number	2.0	0 (default), value in pixels >=0
ScalingMode	Mode of scaling. See "About Scaling" on page 211.	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 211.	Number	2.0	100 (default), percentage value >=0
ScalingWidth	Width of scaled window. See "About Scaling" on page 211.	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

Name	Description	Type	Ver.	Valid Value(s)
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
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	
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	
SSLCACert0, SSLCACert1... SSLCACertN	CA certificates.	String	2.2	
SSLCiphers	SSL ciphers.	String	2.2	ALL, COM, and GOV
SSLCommonName	SSL common name.	String	2.2	
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)
Start	Start the client session immediately.	Boolean	2.0	TRUE, FALSE (default)
Title	Session window title.	String	2.0	"Server" (default)
TCPBrowserAddress	TCP/IP browser address. Used for application/server-name resolution.	String	2.0	
TextColor	Color of text for ICA Client Object specified in #RRGGBB format.	Number	2.0	#000000=black (default) #FFFFFF=white other "#RRGGBB" values...
TransportDriver	Transport used for connection.	String	2.0	TCP/IP (default), IPX, NetBIOS

Name	Description	Type	Ver.	Valid Value(s)
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
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	Username for authentication.	String	2.0	
Version	Version of ICA Client Object.	Read-only string	2.0	Current ICO 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	
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	
Width	Width of ICA Client Object (read-only).	Read-only number	2.0	
WinstationDriver	Winstation driver.	String	2.0	"ICA 3.0" (default)
WorkDirectory	Default starting directory on server.	String	2.0	
XmlAddressResolutionType	Address resolution used for XML requests.	String	2.2	"DNS-Port" (default), "IPv4-Port"
EnableSessionSharingHost	Enable session sharing host mode.	Boolean	2.3	FALSE(default), TRUE

Name	Description	Type	Ver.	Valid Value(s)
IPCLaunch	Use Inter Process Communication (IPC) to launch session.	Boolean	2.3	TRUE(default), FALSE
LongCommandLine	Parameters for InitialProgram published application.	String	2.3	
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	
TWIDisableSessionSharing	Disable session sharing for seamless sessions.	Boolean	2.3	FALSE(default)

ICA Browser Name Resolution

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

The ICA 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.

Name	Description	Type	Possible Value(s)
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	
LocIPXBrowserAddress	Local IPX Browser address. Used for application/server-name resolution.	String	
LocNetbiosBrowserAddress	Local NetBIOS Browser address. Used for application/server-name resolution.	String	
LocTCPBrowserAddress	Local TCP/IP Browser address. Used for application/server-name resolution.	String	

Usage

DoNotUseDefaultCSL controls which set of addresses to use.

When set to FALSE, which is the default, the HTTPBrowserAddress, IPXBrowserAddress, NetbiosBrowserAddress, and TCPBrowserAddress properties are used. When set to TRUE, the LocHTTPBrowserAddress, LocIPXBrowserAddress, LocNetbiosBrowserAddress, and LocTCPBrowserAddress properties are used.

The difference between the two is that those starting with “Loc” are considered to be strong overrides and the client will 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. “Loc” addresses are more specific and there is less room for confusion about name resolution.

BrowserProtocol determines which protocol to use to contact the ICA Browser. For IPX, SPX, and NetBIOS this is fairly straightforward. Both IPX and SPX use IPXBrowserAddress or LocIPXBrowserAddress. NetBIOS uses NetbiosBrowserAddress or LocNetbiosBrowserAddress.

UDP and HTTPonTCP both 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 very difficult to publish applications on the Internet because most firewalls will prohibit UDP traffic.

Most of the time it is perfectly satisfactory to use the non-“Loc” browser addresses. The only time it becomes a serious issue is when there are previous installations of the Citrix ICA 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 ICA Client.

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

If you need to write a logon script, use a form to include fields like username, password, and domain name. Submit these values to the ICA Client Object using SetProp. Don't 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's released.

About Scaling

Scaling properties, ScalingMode, ScalingPercent, ScalingHeight, and ScalingWidth, can be used to determine the initial "scaled" state of the session.

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 will be 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 ICO 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 fifty percent of the control window.
2	Size	This setting instructs ICO 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 ICO 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 will 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 56 in this book.

Example Scripts

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

This chapter contains examples of the following types:

- Embedding
- Scripting
- Logon Methods
- Custom User Interfaces
- Mouse Click
- Scaling Methods
- Other Examples
- Session Caching
- Name Enumeration
- Automatic Application Resizing
- Client Network Name and Address
- Error Message Text from Error Codes
- Global Logoff and Disconnects

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 can be found in case 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>
```

Netscape Plug-in

In the following 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.

Events

The following parts of the example below illustrate the event interface.

- An alert box is displayed when the ICA Client connects to the server (OnConnect event).
- An alert box is displayed when the user is successfully logged on to the server (OnLogon events).
- An alert box is displayed when the user is disconnected from the server (OnDisconnect event).

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

ActiveX

```
<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"
type="text/javascript">

<!--
    icaobj_OnDisconnect();
//-->
```

<contd ...>

<contd ...>

```
</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>
```

Netscape Plug-in

```
<html>
<head>
<title>Event Example for Netscape 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 Netscape 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 is initiated on the server with the specified credentials.

ActiveX

```
<html>
<head><title>Login 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>Login 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>
```

Netscape Plug-in

```
<html>
<head>
<title>Login Example for Netscape 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>Login Example for Netscape Plugin</h1>
<p>Login 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 ICA Client session prior to connection.

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

Important The `UIActive` property must be set to `true` for `CustomMessage` to work.

ActiveX

```
<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>
```

Netscape Plug-in

```
<html>
<head>
<title>Custom UI Example for Netscape Plugin</title>
</head>
<body>
<center>
<h1>Custom UI Example for Netscape 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 The UIActive property must be set to true to enable the OnClick event.

ActiveX

```
<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 fol-
lowing 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>
```

Netscape Plug-in

```
<html>
<head>
<title>UI OnClick Example for Netscape 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 Netscape 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 Netscape Plug-in.

ScaleEnable and ScaleDisable

ActiveX

```
<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>
```

Netscape Plug-in

```
<html>
<head>
  <title>ScaleEnable/ScaleDisable</title>
</head>
<body>
<h1>ScaleEnable/ScaleDisable</h1>
<center>
<br>
<embed name=IC01
type=application/x-ica
width=640 height=480
Address=MyServer
DesiredHRES=800 DesiredVRES=600>
<br>
<form>
<input type="button" value="ScaleEnable"
onClick="document.IC01.ScaleEnable()">
<input type="button" value="ScaleDisable"
onClick="document.IC01.ScaleDisable()">
</form>
</center>
</body>
</html>
```

ScaleUp and ScaleDown

ActiveX

```
<html>
<head>
  <title>ScaleUp/ScaleDown</title>
</head>
<body>
<h1>ScaleUp/ScaleDown</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3" id=IC01
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.IC01.ScaleDown()">
<input type="button" value="ScaleUp"
onClick="document.IC01.ScaleUp()">
</form>
</center>
</body>
</html>
```

Netscape Plug-in

```
<html>
<head>
  <title>ScaleUp/ScaleDown</title>
</head>
<body>
<h1>ScaleUp/ScaleDown</h1>
<center>
<br>
<embed name=IC01
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.IC01.ScaleDown()">
<input type="button" value="ScaleUp"
onClick="document.IC01.ScaleUp()">
</form>
</center>
</body>
</html>
```

ScalePercent

ActiveX

```
<html>
<head>
  <title>ScalePercent</title>
</head>
<body>
<h1>ScalePercent</h1>
<center>
<br>
<object
classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3" id=IC01
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.IC01.ScalePercent
(parseInt(this.form.percent.value))">
<br>
</form>
</center>
</body>
</html>
```


Netscape Plug-in

```
<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

ActiveX

```
<html>
<head>
  <title>ScaleToFit</title>
</head>
<body>
<h1>ScaleToFit</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
id=IC01 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.IC01.ScaleToFit()">
</form>
</center>
</body>
</html>
```

Netscape Plug-in

```
<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

ActiveX

```
<html>
<head>
  <title>ScaleDialog</title>
</head>
<body>
<h1>ScaleDialog</h1>
<center>
<br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
id=IC01 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.IC01.ScaleDialog()">
</form>
</center>
</body>
</html>
```

Netscape Plug-in

```
<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 Netscape Plug-In and ActiveX respectively, scaling properties of a session are set using various scaling modes and associated scaling properties.

Example 1

In the following example, `ScalingMode=0`; that is, session scaling is disabled.

ActiveX

```
<html>
<head></head>
<body>
<h1>Embed with ScalingMode=0 (Disabled)</h1>
<center><br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
        id=IC01 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>
```

Netscape Plug-in

```
<html>
<head></head>
<body>
<h1>Embed with DesiredHRES=800 DesiredVRES=600 width=640 height=480
ScalingMode=0 (Disabled)</h1>
<center><br>
<embed name=IC01
        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`; that is, scaling size is specified in percentage values. The percentage by which the session is scaled is specified by `ScalingPercent`.

ActiveX

```
<html>
<head></head>
<body>
<h1>Embed with ScalingMode=1 ScalingPercent=50</h1>
<center><br>
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
        id=IC01 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>
```

Netscape Plug-in

```
<html>
<head></head>
<body>
<h1>Embed with DesiredHRES=800 DesiredVRES=600 width=640 height=480
ScalingMode=1 ScalingPercent=50</h1>
<center> <br>
<embed name=IC01
        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

```
<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>
```

Netscape Plug-in

```
<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

```
<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>
```

Netscape Plug-in

```
<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>
```

Other Examples

ActiveX

The following example illustrates how scripting is used in client-side HTML/VBScript to allow different applications to be started by the user:

- 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 logoff from the server.
- Use the About button to call the icaObj>AboutBox method to display information about the currently installed ICA Client for 32-bit 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 re-enabled 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 cellpadding="1" cellspacing="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>

```

Netscape Plug-In

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

- 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 via 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 logoff from the server.
- Use the About button to call the icaObj.AboutBox method to display information about the currently installed ICA Client for 32-bit Windows.

```
<html>
<head>
<title>ICA Client Object sample - scripting Netscape 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)
{
    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()
}
```

<contd ...>

<contd ...>

```
// -->
</script>
</head>
<body>
<center>
<form name="appsForm">
<table cellpadding="1" cellspacing="10" border="0">
<tr><td align="center">
    <embed
        name="icaObj"
        type="application/x-ica"
        width="320" height="240">
</td></tr>

<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

The following 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 =
IC01.AttachSession(form.AttachSessionId.value)
    form.AttachClientRC.value = IC01.GetLastClientError()
}
function Detach(form)
{
    form.DetachRC.value =
IC01.DetachSession(form.DetachSessionId.value)
    form.DetachClientRC.value = IC01.GetLastClientError()
}
function IsAttached(form)
{
    form.IsAttachedBool.value =
IC01.IsSessionAttached(form.IsAttachedSessionId.value)
    form.IsAttachedClientRC.value = IC01.GetLastClientError()
}
function IsDetached(form)
{
    form.IsDetachedBool.value =
IC01.IsSessionDetached(form.IsDetachedSessionId.value)
    form.IsDetachedClientRC.value = IC01.GetLastClientError()
}
function IsRunning(form)
{
    form.IsRunningBool.value =
IC01.IsSessionRunning(form.IsRunningSessionId.value)
    form.IsRunningClientRC.value = IC01.GetLastClientError()
}
```

<contd ...>

<contd ...>

```
function SetId(form)
{
    form.SetSessionIdRC.value =
IC01.SetSessionId(form.SetSessionId.value)
    form.SetSessionIdClientRC.value = IC01.GetLastClientError()
}
function SetAddress(form)
{
    IC01.SetProp("Address", form.Address.value)
}
function SetSessionIdProp(form)
{
    IC01.SetProp("SessionId", form.SessionId.value)
}
function SetSessionCacheEnable(form)
{
    IC01.SetProp("SessionCacheEnable",
form.SessionCacheEnable.value)
}
function MyStartup()
{
    document.IC01.Startup()
    SetAddress(SessionCache)
    SetSessionIdProp(SessionCache)
    SetSessionCacheEnable(SessionCache)
}
// -->
</script>
</head>
<body onLoad="MyStartup()">
<h1>Session Cache example</h1>
<center>
<object
    id="IC01"
    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 ...>

```

<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 ...>

<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>
```


<contd...>

```

function Enum(hndEnum)
{
    var EnumCnt
    var i
    var EnumName
    if(hndEnum != 0)
    {
        EnumCnt = document.IC01.GetEnumNameCount(hndEnum)
        if(EnumCnt == 0)
        {
            alert("No names to enumerate")
        }
        for(i = 0; i < EnumCnt; i++)
        {
            EnumName = document.IC01.GetEnumNameByIndex(hndEnum,i)
            alert("index "+i+" Name "+EnumName)
        }
        document.IC01.CloseEnumHandle(hndEnum)
    }
    else
    {
        alert("did not get enum handle")
    }
}

function MyStartup()
{
    document.IC01.Startup()
}

//-->
</script>
</head>
<body onLoad="MyStartup()">
<h1>Enumeration Example</h1>

<center>
<br>
<object
    id="IC01"
    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

The following 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 language="javascript" for="ICOl"
event="OnChannelDataReceived(ChannelName)"
  type="text/javascript">
    <!--
    ICOl_OnChannelDataReceived(ChannelName);
    //-->
  </script>
<script language="JavaScript">
  <!--
  function ICOl_OnConnect()
  {
    document.results.ChannelName.value = ""
    document.results.PktCount.value = 0
    document.results.TotalPktSize.value = 0
  }
  function ICOl_OnDisconnect()
  {
    document.results.ChannelName.value = ""
    document.results.PktCount.value = 0
    document.results.TotalPktSize.value = 0
  }
  function ICOl_OnChannelDataReceived(ChannelName)
  {
    var DataType
    var DataSize
    var ChannelData
    var rc
    var ClientRC

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

<contd...>

<contd...>

```

{
    ChannelData = document.IC01.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.IC01.SendChannelData(ChannelName, ChannelData,
    DataSize, DataType)
    ClientRC = document.IC01.GetLastClientError()
    document.results.VCForm.value = rc
    document.results.CltErr.value = ClientRC
    }
    function WriteData(form)
    {
        form.VCForm.value =
    document.IC01.SendChannelData(form.SendChannel.value,
    form.SendData.value,0,0)
        form.CltErr.value = document.IC01.GetLastClientError()
    }
    function GetChannelInfo(form)
    {
        var cnt
        var i
        var ChannelName
        var ChannelNum
        cnt = document.IC01.GetChannelCount()
        alert("Number of channels : " + cnt)
        for(i=0; i < cnt ; i++)
        {
            ChannelName = document.IC01.GetChannelName(i)
            ChannelNum = document.IC01.GetChannelNumber(ChannelName)
            alert("Index "+ i + " ChannelName " + ChannelName +
            " Number " + ChannelNum)
        }
    }
}

```

<contd...>

<contd...>

```

function GetGlobalChannelInfo(form)
{
    var cnt
    var i
    var ChannelName
    var ChannelNum
    cnt = document.IC01.GetGlobalChannelCount()
    alert("Number of channels : " + cnt)
    for(i=0; i < cnt ; i++)
    {
        ChannelName = document.IC01.GetGlobalChannelName(i)
        ChannelNum = document.IC01.GetGlobalChannelNumber(Channel-
Name)

        alert("Index " + i + " ChannelName " + ChannelName +
" Number " + ChannelNum)
    }
}
function GetMaxChannelInfo(form)
{
    var cnt,cbRead,cbWrite
    cnt = document.IC01.GetMaxChannelCount()
    cbRead = document.IC01.GetMaxChannelRead()
    cbWrite = document.IC01.GetMaxChannelWrite()
    alert("Number of channels : " + cnt + " Max Read: "+cbRead+
" Max Write: "+cbWrite)
}
function GetChFlags(form)
{
    var Flags
    Flags = document.IC01.GetChannelFlags(form.SendChannel.value)
    alert("Channel Flags: " + Flags)
}
function SetChFlags(form)
{
    var rc
    rc = document.IC01.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:;<=>?@ABCDEFGHIJKLMN O PQRSTU-
VWXYZ[\]^_`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.IC01.GetMaxChannelWrite()
    cbLen = GetRandomNumber(1,cbMax)
    RndStr = GetRandomString(cbLen)
    rc = document.IC01.SendChannelData(ChannelName, RndStr, cbLen,
2)
        return(rc)
    }
    function MyStartup()
    {
        document.IC01.Startup()
    }
    //-->
</script>
</head>
<body onLoad="MyStartup()">
<h1>Examples of Virtual Channel methods</h1>

<center>
<br>
<object
    id="IC01"
    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=ClErr
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

The following 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="OnWindow-
Sized(WndType, Width,Height)"
    type="text/javascript">
    <!--
    ICO1_OnWindowSized(WndType, Width,Height);
    //-->
  </script>

  <script language="javascript" for="ICO1" event="OnWindow-
Moved(WndType, 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="IC01" event="OnWindowDe-
stroyed(WndType)"
    type="text/javascript">
    <!--
    IC01_OnWindowDestroyed(WndType);
    //-->
</script>
<script language="javascript" for="IC01" event="OnWindowCloseRe-
quest()"
    type="text/javascript">
    <!--
    IC01_OnWindowCloseRequest();
    //-->
</script>
<script language="JavaScript">
    <!--
    function MyStartup()
    {
        document.IC01.Startup()
        GetSize(document.results)
        GetPosition(document.results)
        document.IC01.SetProp("Title", "My session")
        document.IC01.SetProp("ControlWindowText", "Undocked")
    }
    function SetPos(form)
    {
        document.IC01.SetWindowPosition(parseInt(form.WndType.value),
        parseInt(form.xPos.value), parseInt(form.yPos.value),
        parseInt(form.WndFlags.value))
    }
    function SetSize(form)
    {
        document.IC01.SetWindowSize(parseInt(form.WndType.value),
        parseInt(form.Width.value), parseInt(form.Height.value),
        parseInt(form.WndFlags.value))
    }
    function Show(form)
    {
        document.IC01.DisplayWindow(parseInt(form.WndType.value))
    }

```

<contd...>

<contd...>

```
function Hide(form)
{
    document.IC01.HideWindow(parseInt(form.WndType.value))
}
function GetPosition(form)
{
    form.xPos.value = document.IC01.GetWindowXPosition(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
    form.yPos.value = document.IC01.GetWindowYPosition(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
}
function GetSize(form)
{
    form.Width.value = document.IC01.GetWindowWidth(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
    form.Height.value = document.IC01.GetWindowHeight(parseInt(form.WndType.value), parseInt(form.WndFlags.value))
}
function Undock(form)
{
    document.IC01.UndockWindow()
}
function Dock(form)
{
    document.IC01.DockWindow()
}
function OnTop(form)
{
    document.IC01.PlaceWindowOnTop()
}
function OnBottom(form)
{
    document.IC01.PlaceWindowOnBottom()
}
function Maximize(form)
{
    document.IC01.MaximizeWindow()
}
```

<contd...>

<contd...>

```
function Minimize(form)
{
    document.IC01.MinimizeWindow()
}
function Restore(form)
{
    document.IC01.RestoreWindow()
}
function ScaleToFit(form)
{
    document.IC01.ScaleToFit()
}
function GetDesktopInfo(form)
{
    form.Width.value = document.IC01.GetSessionWidth()
    form.Height.value = document.IC01.GetSessionHeight()
    form.Color.value = document.IC01.GetSessionColorDepth()
}
function GetScreenInfo(form)
{
    form.Width.value = document.IC01.GetScreenWidth()
    form.Height.value = document.IC01.GetScreenHeight()
    form.Color.value = document.IC01.GetScreenColorDepth()
}
function IC01_OnWindowSized(WndType, width, height)
{
    document.results.WndType.value = WndType
    document.results.Width.value = width
    document.results.Height.value = height
}
function IC01_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.ICOl.SetProp('AutoScale','ON')"></td>
<td align=center><input type="button" value="AutoScale(OFF)"
onClick="document.ICOl.SetProp('AutoScale','OFF')"></td>
</tr>
<tr>
<td align=center><input type="button" value="ShowTitleBar"
onClick="document.ICOl.ShowTitleBar()"></td>
<td align=center><input type="button" value="HideTitleBar"
onClick="document.ICOl.HideTitleBar()"></td>
</tr>
<tr>
<td align=center><input type="button" value="EnableSizingBorder"
onClick="document.ICOl.EnableSizingBorder()"></td>
<td align=center><input type="button" value="DisableSizingBorder"
onClick="document.ICOl.DisableSizingBorder()"></td>
</tr>

```

<contd...>

<contd...>

```

<tr>
<td align=center><input type="button" value="FullScreenWindow"
onClick="document.IC01.FullScreenWindow()"></td>
<td align=center><input type="button" value="FocusWindow"
onClick="document.IC01.FocusWindow()"></td>
</tr>
<tr>
<td align=center><input type="button" value="AutoAppResize(ON)"
onClick="document.IC01.SetProp('AutoAppResize', 'ON')"></td>
<td align=center><input type="button" value="AutoAppResize(OFF)"
onClick="document.IC01.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="doc-
ument.IC01.NewWindow(0,0,0,0,0)"></td>
<td align=center><input type="button" value="DeleteWindow"
onClick="document.IC01.DeleteWindow()"></td>
</tr>
<tr>
<td align=center><input type="button" value="EnableKeyboard"
onClick="document.IC01.EnableKeyboardInput()"></td>
<td align=center><input type="button" value="DisableKeyboard"
onClick="document.IC01.DisableKeyboardInput()"></td>
</tr>
<tr>
<td align=center><input type="button" value="EnableMouse"
onClick="document.IC01.EnableMouseInput()"></td>
<td align=center><input type="button" value="DisableMouse"
onClick="document.IC01.DisableMouseInput()"></td>
</tr>
</table>
</form>
</div>
</center>
</body>
</html>

```

Client Network Name and Address

The following 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.IC01.GetClientNetworkName()
      }
      function GetAddrCnt(form)
      {
        form.addrcnt.value = document.IC01.GetClientAddressCount()
      }
      function GetAddr(form)
      {
        form.addr.value = document.IC01.GetClientAddress(0)
      }
      function MyStartup()
      {
        document.IC01.Startup()
      }
    //-->
  </script>
</head>
<body onLoad="MyStartup()">
<h1>Client address example</h1>
<center>
<br>
<object
  id="IC01"
  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

The following example illustrates the use of methods available to extract error message text if the ICO error code(s) is 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

The following 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="IC01" event="OnConnect()"
type="text/javascript">
        <!--
            IC01_OnConnect();
        //-->
    </script>
    <script language="javascript" for="IC01" event="OnDiscon-
nect()" type="text/javascript">
        <!--
            IC01_OnDisconnect();
        //-->
    </script>
    <script language="javascript" for="IC01" event="OnConnect-
Failed()" type="text/javascript">
        <!--
            IC01_OnConnectFailed();
        //-->
    </script>
    <script language="javascript" for="IC01" event="OnDisconnect-
Failed()" type="text/javascript">
        <!--
            IC01_OnDisconnectFailed();
        //-->
    </script>
    <script language="javascript" for="IC01" event="OnDisconnect-
Sessions(hCommand)" type="text/javascript">
        <!--
            IC01_OnDisconnectSessions(hCommand);
        //-->
    </script>
    <script language="javascript" for="IC01" event="OnDisconnect-
SessionsFailed(hCommand)" type="text/javascript">
        <!--
            IC01_OnDisconnectSessionsFailed(hCommand);
        //-->
    </script>
    <script language="javascript" for="IC01" event="OnLogoffSes-
sions(hCommand)" type="text/javascript">
```

<contd...>

<contd...>

```

<!--
    ICO1_OnLogoffSessions (hCommand) ;
//-->
</script>
<script language="javascript" for="ICO1" event="OnLogoffSessionsFailed(hCommand)" type="text/javascript">
<!--
    ICO1_OnLogoffSessionsFailed(hCommand) ;
//-->
</script>
<script language="javascript" for="ICO1" event="OnSessionSwitch(hOldSession, hNewSession)" type="text/javascript">
<!--
    ICO1_OnSessionSwitch(hOldSession, hNewSession);
//-->
</script>
<script language="javascript" for="ICO1" event="OnSessionEventPending(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="OnPublishedApp()" type="text/javascript">
<!--
    ICO1_OnPublishedApp();
//-->
</script>
<script language="javascript" for="ICO1" event="OnPublishedAppFailed()" 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 =
        IC01.GetLastError()
        document.GlobalLogoff.LogoffSessionsClientRC.value =
        IC01.GetLastClientError()
    }
    function IC01_OnLogoffSessionsFailed(hCommand)
    {
        alert("OnLogoffSessionsFailed " + hCommand)
        document.GlobalLogoff.LogoffSessionshCmd.value = hCom-
mand
        document.GlobalLogoff.LogoffSessionsRC.value =
        IC01.GetLastError()
        document.GlobalLogoff.LogoffSessionsClientRC.value =
        IC01.GetLastClientError()
    }
    function IC01_OnSessionSwitch(hOldSession, hNewSession)
    {
        alert("OnSessionSwitch Old Session:" + hOldSession + "
New Session: " + hNewSession)
    }
    function IC01_OnSessionEventPending(hSession, iEvent)
    {
        alert("OnSessionEventPending Session:" + hSession + "
Event: " + iEvent)
    }
    function DisconnectSessions(form)
    {
        form.DisconnectSessionshCmd.value = IC01.Disconnect-
Sessions(form.DisconnectGroupId.value)
        form.DisconnectSessionsRC.value = IC01.GetLastError()
        form.DisconnectSessionsClientRC.value = IC01.GetLast-
ClientError()
    }
    function LogoffSessions(form)
    {
        form.LogoffSessionshCmd.value = IC01.LogoffSes-
sions(form.LogoffGroupId.value)
        form.LogoffSessionsRC.value = IC01.GetLastError()
        form.LogoffSessionsClientRC.value = IC01.GetLastClien-
tError()
    }
    function SetSessionGroupId(form)
    {
        form.SetSessionGroupIdRC.value = IC01.SetSession-
GroupId(form.SetGroupId.value)
        form.SetSessionGroupIdClientRC.value = IC01.GetLast-
ClientError()
    }

```

<contd...>

<contd...>

```

function SwitchSession(form)
{
    form.SwitchRC.value      = ICO1.SwitchSession(form.SwitchhSession.value)
    form.SwitchClientRC.value = ICO1.GetLastClientError()
}
function GetSessionHandle(form)
{
    form.GetSessionHandlehSession.value = ICO1.GetSessionHandle()
    form.GetSessionHandleRC.value       = ICO1.GetLastError()
    form.GetSessionHandleClientRC.value = ICO1.GetLastClientError()
}
function GetSessionCount(form)
{
    form.GetSessionCountNumber.value    = ICO1.GetSessionCount()
    form.GetSessionCountRC.value        = ICO1.GetLastError()
    form.GetSessionCountClientRC.value  = ICO1.GetLastClientError()
}
function GetSessionHandleByIndex(form)
{
    form.GSHBI_hSession.value = ICO1.GetSessionHandleByIndex(form.GSHBI_Index.value)
    form.GSHBI_RC.value       = ICO1.GetLastError()
    form.GSHBI_ClientRC.value = ICO1.GetLastClientError()
}
function GetSessionGroupCount(form)
{
    form.GSGC_Number.value    = ICO1.GetSessionGroupCount(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.SessionExit-
TimeoutProp.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.DesiredHRe-
sProp.value)
}
function SetDesiredVResProp(form)
{
    ICO1.SetProp("DesiredVRes", form.DesiredVRe-
sProp.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...>
        IC01.SetProp("EnableSessionSharingClient", Value)
    }
    function SetESSHProp(form)
    {
        var Value
        // enable or disable session sharing host support
        if(form.ESSHOption.checked)
        {
            Value = "TRUE"
        }
        else
        {
            Value = "FALSE"
        }
        IC01.SetProp("EnableSessionSharingHost", Value)
    }
    function SetSSNProp(form)
    {
        IC01.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"
        }
        IC01.SetProp("SessionSharingLaunchOnly", Value)
    }
    function SetInitialProgramProp(form)
    {
        IC01.SetProp("InitialProgram", form.InitialProgram-
Prop.value)
    }
    function SetLCLProp(form)
    {
        IC01.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...>

```

<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...>

```

<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 Initial-
Program" 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="IC01.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="Discon-
nectSessions" 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="SetSes-
sionGroupId" 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...>

```

<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="GetSessionHandleCli-
entRC" 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="GetSessionHandle-
ByIndex" 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="GetSessionCountClie-
ntRC" 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 ICO and client errors that occurred as a result of the last request.

An ICA Client Object error is the last error reported for the ICA Client Object; an ICA Client error is the last error obtained from the ICA Client engine. Both ICO and the ICA Client have 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.

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

ICA Client Object Error Codes

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"

Error Number	Defined Name	Ver.	Error String
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"
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"

Error Number	Defined Name	Ver.	Error String
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"
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"

Error Number	Defined Name	Ver.	Error String
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."
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"

Error Number	Defined Name	Ver.	Error String
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"
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"

Error Number	Defined Name	Ver.	Error String
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\nURL: %s \n\nIf this Web site is trusted, add it to the Trusted Zone list in Internet Explorer."

Client Error Codes

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"
1024	CLIENT_ERROR_PD_NAME_NOT_FOUND	2.1	"Protocol Driver name not found in MODULE.INI"

Error Number	Defined Name	Version	Error String
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)"

Error Number	Defined Name	Version	Error String
1081	CLIENT_ERROR_MISSING_CONNECTION_SECTION	2.1	"(Internal error - missing connection section)"
1082	CLIENT_ERROR_BAD_COMBINE_ENTRIES	2.1	"(Internal error - CombinePrivateProfileEntries 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)"

Error Number	Defined Name	Version	Error String
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"
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"

Error Number	Defined Name	Version	Error String
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"

License Agreement

This is a legal agreement (AGREEMENT) between you, the licensed user or the authorized representative of the licensed user, and Citrix Systems, Inc. (CITRIX). BY USING THE ATTACHED SOFTWARE DEVELOPMENT KIT, YOU ARE INDICATING THAT YOU ARE AUTHORIZED TO BIND THE LICENSED USER IN CONTRACT AND THAT THE LICENSED USER ACCEPTS THE TERMS OF THIS AGREEMENT. IF YOU, AS THE LICENSED USER OR REPRESENTATIVE OF THE LICENSED USER DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT DOWNLOAD, INSTALL, COPY OR OTHERWISE USE THE CITRIX SOFTWARE DEVELOPMENT KIT.

1. **GRANT OF LICENSE.** The CITRIX software development kit includes certain example source code, tools, utilities, program interfaces and text files (collectively called the “SOFTWARE DEVELOPMENT KIT”) to software that provides services on a computer called a server (“Server Software”) and software that allows a computer to access or utilize the services provided by the Server Software (“Client Software”). CITRIX grants to the licensed user the following limited, non-exclusive and non-transferable rights to the SOFTWARE DEVELOPMENT KIT and accompanying documentation downloaded or otherwise obtained from Citrix:

- a) **Installation and Transfer.** The licensed user may install one copy of the SOFTWARE DEVELOPMENT KIT on a single computer owned by the licensed user. The licensed user may transfer the SOFTWARE DEVELOPMENT KIT 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 SOFTWARE DEVELOPMENT KIT in machine-readable form solely for backup purposes, provided that the licensed user reproduces all proprietary notices on the copy.
- b) **Use of the SOFTWARE DEVELOPMENT KIT.** The licensed user may use the SOFTWARE DEVELOPMENT KIT and accompanying documentation solely to develop applications that access or utilize the Server Software.

c) **Export Notice.** The licensed user may not export or re-export the SOFTWARE DEVELOPMENT KIT in any form without the appropriate United States and foreign government licenses. The licensed user is responsible for maintaining the security of the environment in which the SOFTWARE DEVELOPMENT KIT is used. None of the SOFTWARE DEVELOPMENT KIT or underlying information or technology may be exported or re-exported (i) into (or to a national or resident of) Cuba, Iraq, Libya, Sudan, North Korea, Iran, Syria or any other country to which the United States has embargoed goods; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Denial Orders. By installing or using the SOFTWARE DEVELOPMENT KIT, the licensed user is agreeing to the foregoing and is representing and warranting that such licensed user is not located in, under the control of, or a national or resident of any such country or on any such list. In addition, the licensed user is responsible for complying with any local laws in such licensed user's jurisdiction which may impact such licensed user's right to import, export or use the SOFTWARE DEVELOPMENT KIT.

If the SOFTWARE DEVELOPMENT KIT is identified as a North American or not-for-export product (for example, on the box, media or in the installation process), then, unless the licensed user has an exemption from the United States Department of State, the following applies: EXCEPT FOR EXPORT TO CANADA FOR USE IN CANADA BY CANADIAN CITIZENS, THE SOFTWARE DEVELOPMENT KIT AND ANY UNDERLYING TECHNOLOGY MAY NOT BE EXPORTED OUTSIDE THE UNITED STATES OR TO ANY FOREIGN ENTITY OR "FOREIGN PERSON" AS DEFINED BY U.S. GOVERNMENT REGULATIONS, INCLUDING WITHOUT LIMITATION, ANYONE WHO IS NOT A CITIZEN, NATIONAL OR LAWFUL PERMANENT RESIDENT OF THE UNITED STATES. BY DOWNLOADING OR USING THE SOFTWARE DEVELOPMENT KIT, THE LICENSED USER IS AGREEING TO THE FOREGOING AND THE LICENSED USER IS WARRANTING THAT SUCH LICENSED USER IS NOT A "FOREIGN PERSON" OR UNDER THE CONTROL OF A "FOREIGN PERSON."

d) **Other.** Notice to Users — The licensed user shall inform all users of the SOFTWARE DEVELOPMENT KIT of the terms and conditions of this AGREEMENT. Not For Resale Software - If the SOFTWARE DEVELOPMENT KIT is labeled "Not For Resale" or "NFR," this license only permits use for demonstration, test, or evaluation purposes.

2. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS. The licensed user may not rent, lease, sublicense, assign or otherwise transfer or distribute the SOFTWARE DEVELOPMENT KIT. The licensed user may not modify, translate, reverse engineer, decompile, or disassemble or copy (except for the backup copy of the SOFTWARE DEVELOPMENT KIT) any part of the SOFTWARE DEVELOPMENT KIT, except to the extent such foregoing restriction is expressly prohibited by applicable law. The licensed user may not remove any proprietary notices, labels, or marks on the SOFTWARE DEVELOPMENT KIT and accompanying documentation.

THE LICENSED USER MAY NOT USE, COPY, MODIFY, OR TRANSFER THE SOFTWARE DEVELOPMENT KIT OR ANY COPY IN WHOLE OR IN PART, OR GRANT ANY RIGHTS IN THE SOFTWARE DEVELOPMENT KIT OR ACCOMPANYING DOCUMENTATION, EXCEPT AS EXPRESSLY PROVIDED IN THIS LICENSE. ALL RIGHTS NOT EXPRESSLY GRANTED ARE RESERVED BY CITRIX OR ITS SUPPLIERS.

MAINTENANCE. Citrix is not obligated to provide maintenance, updates or any other support to the licensed user with respect to the SOFTWARE DEVELOPMENT KIT and accompanying documentation licensed under this AGREEMENT.

DISCLAIMER OF WARRANTY. THE SOFTWARE DEVELOPMENT KIT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CITRIX AND ITS SUPPLIERS FURTHER DISCLAIM ALL WARRANTIES OR CONDITIONS, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK ARISING OUT OF THE USE OR PERFORMANCE OF THE SOFTWARE DEVELOPMENT KIT AND ACCOMPANYING DOCUMENTATION REMAINS WITH RECIPIENT. CITRIX does not warrant that the SOFTWARE DEVELOPMENT KIT will be uninterrupted or error free.

SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES SO THE ABOVE EXCLUSIONS MAY NOT APPLY TO THE LICENSED USER. THIS WARRANTY GIVES THE LICENSED USER SPECIFIC LEGAL RIGHTS. THE LICENSED USER MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

PROPRIETARY RIGHTS. This license is not a sale. Title and copyrights to the SOFTWARE DEVELOPMENT KIT and accompanying documentation and any copy made by the licensed user remain with CITRIX or its suppliers.

LIMITATION OF LIABILITY. IN NO EVENT WILL CITRIX OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION, LOSS OF DATA, LOSS OF INCOME, LOSS OF OPPORTUNITY OR PROFITS, BUSINESS INTERRUPTION, COST OF RECOVERY OR OTHER PECUNIARY LOSS OR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF THE SOFTWARE DEVELOPMENT KIT, REFERENCE MATERIALS OR ACCOMPANYING DOCUMENTATION OF THE SOFTWARE DEVELOPMENT KIT, HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY. THIS LIMITATION WILL APPLY EVEN IF CITRIX, ITS SUPPLIERS OR AUTHORIZED DISTRIBUTORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO THE LICENSED USER.

TERMINATION. This AGREEMENT is effective until terminated. The licensed user may terminate this AGREEMENT at any time by removing the SOFTWARE DEVELOPMENT KIT from such licensed user's computer and destroying all copies of it. Unauthorized copying of the SOFTWARE DEVELOPMENT KIT or otherwise failing to comply with the terms and conditions of this AGREEMENT will result in automatic termination of this license and will make available to CITRIX other legal remedies. Upon termination of this AGREEMENT, the license granted herein will terminate and the licensed user must immediately destroy the SOFTWARE DEVELOPMENT KIT and all backup copies thereof.

GOVERNMENT END-USERS. The SOFTWARE DEVELOPMENT KIT and accompanying documentation are "commercial items," developed exclusively at private expense, consisting of "commercial computer software" and "commercial computer software documentation" as such terms are defined in the applicable acquisition regulations. If the SOFTWARE DEVELOPMENT KIT and the documentation are licensed hereunder for distribution to Government End-Users, such SOFTWARE DEVELOPMENT KIT and documentation are licensed (i) only as a commercial item, and (ii) with only those rights as are granted to all other end users pursuant to the terms and conditions of this Agreement. If this Agreement fails to meet a Government End-User's minimum needs or is inconsistent with Federal Procurement law, the licensed user agrees to notify Citrix. The following additional statement applies only to procurements governed by DFARS Subpart 227.4 (1988): "Restricted Rights æ Use, duplication and disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 (1988)."

This AGREEMENT will be governed by the laws of the State of Florida, without reference to conflict of laws principles. In any dispute arising out of this AGREEMENT, the licensed user and CITRIX each consent to the exclusive personal jurisdiction and venue in the State and Federal courts within Broward County, Florida. If either Citrix or the licensed user employs attorneys to enforce any rights arising out of or relating to this Agreement, the prevailing party shall be entitled to recover reasonable attorneys' fees and expenses.

Should you have any questions concerning this AGREEMENT, or if you desire to contact Citrix Systems, Inc. for any reason, please write: Citrix Systems, Inc., Customer Service, 851 W. Cypress Creek Road, Fort Lauderdale, Florida, 33309.



Index

A

- about passwords 211
- about scaling 211
- ActiveX 16
 - Embedding 16
 - Events 17
 - Methods 17
 - Properties 17
- Address 200
- Application 200
- applications
 - see* publishing applications and content
- APPSRVINI 200
- audience 9
- AudioBandwidthLimit 200
- AUTHPassword 200
- AUTHUsername 200
- AutoLogonAllowed 200

B

- BackgroundColor 200
- Border 200
- BorderColor 200
- BrowserProtocol 200
- BrowserRetry 200
- Browser-Specific Properties
 - BrowserProtocol 209
 - DoNotUseDefaultCSL 209
 - LocHTTPBrowserAddress 209
 - LocIPXBrowserAddress 209
 - LocNetbiosBrowserAddress 209
 - LocTCPBrowserAddress 209
- BrowserTimeout 200

C

- CacheICAFile 201
- CDMAAllowed 201
- CDN (Citrix Developer Network) 11
- ClearPassword 201
- client errors 295

- ClientAudio 201
- ClientName 201
- ClientPath 201
- ClientVersion 201
- COMAllowed 201
- Compress 201
- Connected 201
- ConnectionEntry 201
- ControlWindowText 201
- CPMAAllowed 201
- CustomMessage 201

D

- Description 201
- DesiredColor 201
- DesiredHRes 202
- DesiredVRes 202
- DisableCtrlAltDel 202
- documentation
 - submitting comments 11
- Domain 202
- DoNotUseDefaultCSL 202

E

- EnableSessionSharingClient 202
- EnableSessionSharingHost 207
- Encrypt 202
- EncryptionLevelSession 202
- error codes 289
- Events Interface
 - OnChannelDataReceived 176
 - OnClick 162
 - OnConnect 163
 - OnConnectFailed 164
 - OnConnecting 165
 - OnDisconnect 166
 - OnDisconnectFailed 167
 - OnDisconnectSessions 190
 - OnDisconnectSessionsFailed 191
 - OnICAFile 168
 - OnICAFileFailed 169
 - OnInitializing 173
 - OnInitialProp 174

- OnLogoffFailed 175
- OnLogoffSessions 192
- OnLogoffSessionsFailed 193
- OnLogon 162, 170
- OnPublishedApp 171
- OnPublishedAppFailed 172
- OnSessionAttach 196
- OnSessionDetach 197
- OnSessionEventPending 195
- OnSessionSwitch 194
- OnWindowCloseRequest 189
- OnWindowCreated 179
- OnWindowDestroyed 180
- OnWindowDisplayed 188
- OnWindowDocked 181
- OnWindowFullscreened 186
- OnWindowHidden 187
- OnWindowMaximized 184
- OnWindowMinimized 183
- OnWindowMoved 178
- OnWindowRestored 185
- OnWindowSized 177
- OnWindowUndocked 182

F

Features

- session caching 28

G

- Global Logoff and Disconnect
 - sample code 274

- Global logoff and disconnect
 - error codes 25
 - events 25
 - features 22
 - methods 19, 24
 - properties 25
 - sample code 26

H

- Height 202
- Hotkey1Char 202
- Hotkey1Shift 202
- Hotkey10Char 203
- Hotkey10Shift 203
- Hotkey2Char 202
- Hotkey2Shift 202
- Hotkey3Char 202

- Hotkey3Shift 202
- Hotkey4Char 202
- Hotkey4Shift 202
- Hotkey5Char 203
- Hotkey5Shift 203
- Hotkey6Char 203
- Hotkey6Shift 203
- Hotkey7Char 203
- Hotkey7Shift 203
- Hotkey8Char 203
- Hotkey8Shift 203
- Hotkey9Char 203
- Hotkey9Shift 203
- HTTPBrowserAddress 202

I

ICA

- error codes 289

- ICA Clients 13

ICA Browser

- Name Resolution 208

- ICA Client error codes 295

ICA Client Object

- data types 45

- error codes 289

- functional specification 45

- Properties 199

- ICAFile 203

- ICAPortNumber 203

- ICASOCKSProtocolVersion 203

- ICASOCKSProxyHost 203

- ICASOCKSProxyPortNumber 203

- ICASOCKSRFC1929UserName 203

- ICASOCKSTimeout 203

- ICO error codes 289

ICO 2.3

- Connection Performance Improvements 21

- Global logoff and disconnect 21

- new error codes 293

- Support for Security Zones 26

- IconIndex 203

- IconPath 204

- InitialProgram 204

- IPCLaunch 208

- IPXBrowserAddress 204

K

- KeyboardTime 204

L

LanaNumber 204
 Launch 204
 load-time 46
 LocHTTPBrowserAddress 204
 LocIPXBrowserAddress 204
 LocNetbiosBrowserAddress 204
 LocTCPBrowserAddress 204
 LogAppend 204
 LogConnect 204
 LogErrors 204
 LogFile 204
 LogFlush 204
 LogKeyboard 204
 LogReceive 204
 LogTransmit 205
 LongCommandLine 208

M

Methods Interface
 AboutBox 56
 Connect 59
 Disconnect 66
 DisconnectSessions 66
 GetClientIdentification 91
 GetInterfaceVersion 90
 GetLastError 99
 GetNotificationReason 103
 GetScreenHeight 111
 GetScreenWidth 113
 GetSessionColorDepth 114
 GetSessionCount 106
 GetSessionCounter 107
 GetSessionGroupCount 108
 GetSessionHandle 109
 GetSessionHandleByIndex 110
 GetSessionHeight 115
 GetSessionString 107, 112
 GetSessionWidth 116
 GetWindowHeight 117
 GetWindowWidth 118
 GetWindowXPosition 119
 GetWindowYPosition 120
 HideTitleBar 122
 HideWindow 121
 IsConnected 123
 IsKeyboardInputEnabled 124
 IsMouseInputEnabled 125–126
 IsSessionAttached 127

IsSessionDetached 128
 IsSessionRunning 129
 IsWindowDocked 130
 LoadIcaFile 131
 Logoff 131–132
 LogoffSessions 133
 MaximizeWindow 135
 MinimizeWindow 134
 NewWindow 136
 PlaceWindowOnBottom 137
 PlaceWindowOnTop 138
 RestoreWindow 139
 RunPublishedApplication 140
 ScaleDialog 141
 ScaleDisable 142
 ScaleDown 143
 ScaleEnable 144
 ScalePercent 145
 ScaleSize 146
 ScaleToFit 147
 ScaleUp 148
 SendChannelData 150
 SetChannelFlags 151
 SetSessionEndAction 152
 SetSessionGroupId 149
 SetSessionId 153
 SetWindowPosition 154
 SetWindowSize 155
 ShowTitleBar 156
 Startup 157
 SwitchSession 158
 UndockWindow 159

MODULEINI 205
 MouseTimer 205

N

Name Enumeration
 when to use it 34
 name enumeration 34
 NetbiosBrowserAddress 205
 Netscape Plug-In
 Embedding 18
 Events 19
 Methods 18
 Properties 18
 Summary 19

Netscape Plug-Ins 16
NFuse Classic
 see Web Interface
NotificationReason 205

O

OutputMode 208

P

Password 205
passwords
 encrypted and unencrypted 211
PasswordType 205
PersistentCacheEnabled 205
Properties 17
Property Interface 46
 ClearProps 47
 DeleteProp 48
 DeletePropByIndex 49
 GetPropCount 50
 GetPropNameByIndex 51
 GetPropValue 56
 GetPropValueByIndex 53
 ResetProps 52, 54
 SetProp 55
ProtocolSupport 205

R

Reliable 205
Requirements 9
run-time 46

S

scaling 211
ScalingHeight 205
ScalingMode 205
 initial states 212
ScalingMode states 211
ScalingPercent 205
ScalingWidth 205
scripting 16
Scrollbars 206
Session 208
session caching
 client settings 33
 features 28
 methods 32

 properties 32
 terminology 28
 when to use it 33
SessionEndAction 206
SessionExitTimeout 208
SessionGroupId 208
sessions
 see ICA sessions
SessionSharingLaunchOnly 206
SessionSharingName 206
SSLCACert 206
SSLCiphers 206
SSLCommonName 206
SSLEnable 206
SSLNoCACerts 206
SSLProxyHost 206
Start 206
system requirements
 see requirements

T

TCPBrowserAddress 206
TextColor 206
Title 206
training 12
TransportDriver 206
TransportReconnectDelay 207
TransportReconnectEnabled 207
TransportReconnectRetries 207
TWIDisableSessionSharing 208
TWIMode 207

U

UIActive 207
UpdatesAllowed 207
UseAlternateAddress 207
UserName 207

V

Version 207
VirtualChannels 207
VSLAllowed 207

W

WFCLIENTINI 207

What is

- session caching 28

What's New

- in ICO 2.2 19, 21

- in ICO 2.3 21

- name enumeration 34

- session caching 28

Width 207

WinstationDriver 207

WorkDirectory 207

X

XmlAddressResolutionType 207

