

Idc Command Reference Guide  
10g Release 3 (10.1.3.3.0)

March 2007

Idc Command Reference Guide, 10g Release 3 (10.1.3.3.0)  
Copyright © 2007, Oracle. All rights reserved.

Contributing Authors: Will Harris, Jean Wilson

Contributors: Eva Cordes, Rick Petty, Sam White

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

# Table of Contents



## Chapter 1: Overview

About This Guide .....	1-1
------------------------	-----

## Chapter 2: IdcCommand: Java Command Utility

IdcCommand Setup and Execution .....	2-2
Command File .....	2-2
Command File Syntax .....	2-3
Precedence .....	2-4
Special Tags and Characters .....	2-4
Configuration Options .....	2-5
Running IdcCommand .....	2-7
Using the Launcher .....	2-7
Quoting .....	2-9
Computed Settings .....	2-9
Launcher Environment Variables .....	2-12
User Interface .....	2-13
Configuring the Launcher .....	2-14
Configuration File Example .....	2-14
Calling Services Remotely .....	2-17

## Chapter 3: IdcCommandX and IdcCommandUX: ActiveX Command Utilities

IdcCommandUX Overview .....	3-2
Setup .....	3-2
Calling Procedures .....	3-2
Visual Basic .....	3-2
Visual C++ .....	3-3
Executing Services .....	3-3
Calling IdcCommandUX from an Active Server Page .....	3-4

HDA example . . . . .	3-4
Creating a COM Object . . . . .	3-5
Initializing the Connection . . . . .	3-5
Defining the Service and Parameters . . . . .	3-5
Referencing Custom Resources . . . . .	3-6
Executing the Service . . . . .	3-6
Retrieving Results . . . . .	3-6
SOAP Example . . . . .	3-7
Formatting Content with a Resource Include . . . . .	3-9
Connecting to Content Server from a Remote Machine . . . . .	3-10
Coding the ASP Page . . . . .	3-11
IdcCommandUX Methods . . . . .	3-16
addExtraHeadersForCommand . . . . .	3-17
closeServerConnection . . . . .	3-17
computeNativeFilePath . . . . .	3-18
computeURL . . . . .	3-19
computeWebFilePath . . . . .	3-22
connectToServer . . . . .	3-23
executeCommand . . . . .	3-24
executeFileCommand . . . . .	3-25
forwardRequest . . . . .	3-26
getLastErrorMessage . . . . .	3-26
init (deprecated) . . . . .	3-27
initRemote . . . . .	3-28

## Chapter 4: IdcClient OCX Component

IdcClient OCX Description . . . . .	4-2
General Description . . . . .	4-2
Events, Methods, and Properties . . . . .	4-3
OCX Events . . . . .	4-3
OCX Methods . . . . .	4-3
OCX Properties . . . . .	4-4
IdcClient OCX Interface . . . . .	4-4
IdcClient OCX Control Setup . . . . .	4-5
Component Setup . . . . .	4-5
Creating a Visual Interface . . . . .	4-5
IdcClient Events . . . . .	4-16
IntradocBeforeDownload . . . . .	4-17
IntradocBrowserPost . . . . .	4-17
IntradocBrowserStateChange . . . . .	4-17
IntradocRequestProgress . . . . .	4-18

IntradocServerResponse . . . . .	4-18
IdcClient Methods . . . . .	4-18
AboutBox . . . . .	4-19
Back . . . . .	4-19
CancelRequest . . . . .	4-20
DoCheckoutLatestRev . . . . .	4-20
DownloadFile . . . . .	4-21
DownloadNativeFile . . . . .	4-21
Drag . . . . .	4-22
EditDocInfoLatestRev . . . . .	4-23
Forward . . . . .	4-23
GoCheckinPage . . . . .	4-24
Home . . . . .	4-25
InitiateFileDownload . . . . .	4-25
InitiatePostCommand . . . . .	4-26
Move . . . . .	4-26
Navigate . . . . .	4-27
NavigateCgiPage . . . . .	4-27
RefreshBrowser . . . . .	4-28
SendCommand . . . . .	4-28
SendPostCommand . . . . .	4-28
SetFocus . . . . .	4-29
ShowDMS . . . . .	4-30
ShowDocInfoLatestRev . . . . .	4-30
ShowWhatsThis . . . . .	4-30
StartSearch . . . . .	4-31
Stop . . . . .	4-31
UndoCheckout . . . . .	4-32
ViewDocInfo . . . . .	4-32
ViewDocInfoLatestRev . . . . .	4-33
ZOrder . . . . .	4-33
IdcClient Properties . . . . .	4-34
ClientControlledContextValue . . . . .	4-34
HostCgiUrl . . . . .	4-34
Password . . . . .	4-35
UseBrowserLoginPrompt . . . . .	4-35
UseProgressDialog . . . . .	4-35
UserName . . . . .	4-35
WorkingDir . . . . .	4-36

## **Appendix A: Third Party Licenses**

Apache Software License . . . . .	A-1
W3C® Software Notice and License . . . . .	A-2
Zlib License . . . . .	A-3
General BSD License. . . . .	A-4
General MIT License . . . . .	A-5
Unicode License. . . . .	A-5
Miscellaneous Attributions . . . . .	A-7

# OVERVIEW

## INTRODUCTION

---

The information contained in this guide is based on Content Server 10gR3. The information is subject to change as the product technology evolves and as hardware and operating systems are created and modified.

Due to the technical nature of browsers, web servers, and operating systems, Oracle, Inc., cannot warrant compatibility with all versions and features of third-party products.

This section contains these topics:

- ❖ [About This Guide](#) (page 1-1)

## ABOUT THIS GUIDE

---

This guide provides information on using the IdcCommand and IdcCommandX utilities to access content server services from other applications. You can access these utilities on the support site.

- ❖ The *IdcCommand Java Command Utility* is a stand-alone Java application that enables users to execute content server services. See [Chapter 2 \(IdcCommand: Java Command Utility\)](#).
- ❖ *IdcCommandX* and IdcCommand UX are ActiveX controls that allows a program to execute a service and retrieve file path information. IdcCommandX serves as a COM wrapper for the standard IdcCommand services used by content server. IdcCommandUX is an updated IdcCommandX control that works with multibyte

languages. See [Chapter 3 \(\*IdcCommandX and IdcCommandUX: ActiveX Command Utilities\*\)](#).

- ❖ An Object Linking and Embedding Control Extension (OCX) control is also provided for connecting to a remote content server and executing Content Server services. The IdcClient OCX control is used within a Windows Visual Basic development environment to gain access to the content and content management functions within Content Server. See [Chapter 4 \(\*IdcClient OCX Component\*\)](#).



**Note:** The information contained in this guide is based on Content Server 10gR3. The information is subject to change as the product technology evolves and as hardware and operating systems are created and modified. Due to the technical nature of browsers, databases, web servers, and operating systems, Oracle, Inc. cannot warrant compatibility with all versions and features of third-party products.



**Note:** This reference guide is part of the Software Developer's Kit (SDK). See *Getting Started with the Software Developer's Kit (SDK)* for more information.



**Note:** For information on using services in custom components, see the *Services Reference Guide* and the *Working with Components* guide.

## Audience

---

This guide is intended for application developers who need to access Content Server functions. This guide provides information on the Java Command Utility, ActiveX Command Utility, and OCX Component for the content server.

This guide is intended for content server administrators who want to use the Layout Manager functionality to provide alternate interface navigation and design.

## Conventions





---

The following conventions are used throughout this guide:

- ❖ The notation `<install_dir>/` is used throughout this guide to refer to the location on your system where Content Server product is installed.
- ❖ Forward slashes (/) are used to separate the directory levels in a path name. A forward slash will always appear after the end of a directory name.



❖ Notes, technical tips, important notices, and cautions use these conventions:

Symbols	Description
	This is a note. It is used to bring special attention to information.
	This is a technical tip. It is used to identify information that can be used to make your tasks easier.
	This is an important notice. It is used to identify a required step or required information.
	This is a caution. It is used to identify information that might cause loss of data or serious system problems.

## Overview

# IDCCOMMAND: JAVA COMMAND UTILITY

## INTRODUCTION

---

The IdcCommand utility is a stand-alone Java application that executes Content Server services. Almost any action you can perform from the content server browser interface or administration applets can be executed from IdcCommand.

The program reads a [Command File](#) (page 2-2), which contains service commands and parameters, and then calls the specified services. A log file can record the time that the call was executed, whether the service was successfully executed, and if there were execution errors.

This section covers the following topics:

- ❖ [IdcCommand Setup and Execution](#) (page 2-2)
- ❖ [Command File](#) (page 2-2)
- ❖ [Running IdcCommand](#) (page 2-7)
- ❖ [Using the Launcher](#) (page 2-7)
- ❖ [Calling Services Remotely](#) (page 2-17)



**Note:** The IdcCommand utility only returns information about the success or failure of the command. To retrieve information from the content server in an interactive session, use the IdcCommandX Java COM wrapper available on Windows platforms. See [Chapter 3 \(IdcCommandX and IdcCommandUX: ActiveX Command Utilities\)](#) for additional information.

## IDCCOMMAND SETUP AND EXECUTION

---

To set up IdcCommand, you must specify the following two things:

- ❖ A [Command File](#) (page 2-2), which specifies the services to be executed and any service parameters.
- ❖ [Configuration Options](#) (page 2-5), which specify the command file and other IdcCommand information. You can set IdcCommand configuration options in two places:
  - In a configuration file, using name/value pairs such as:

```
IdcCommandFile=newfile.hda
IdcCommandUserName=sysadmin
IdcCommandLog=C:/stellent/newlog.txt
ConnectionMode=server
```
  - On the command line when running IdcCommand, specifying option flags such as:

```
-f newfile.hda -u admin -l C:/stellent/newlog.txt -c server
```



**Note:** Command line configuration options override the settings in the configuration file.

### *IdcCommand Execution*

IdcCommand is run from a command line. You can specify the [Configuration Options](#) (page 2-5) either from the command line or in a configuration file. See [Running IdcCommand](#) (page 2-7) for more information.

## COMMAND FILE

---

The command file defines the service commands and parameters that are executed by the IdcCommand utility. The command file follows these rules:

- ❖ [Command File Syntax](#) (page 2-3)
- ❖ [Precedence](#) (page 2-4)
- ❖ [Special Tags and Characters](#) (page 2-4)

## Command File Syntax

---

The command file uses the HDA (hyperdata file) syntax to define service commands.

- ❖ Each service to be executed, along with its parameters, is specified in a `@Properties LocalData` section.
- ❖ For some services, a `@ResultSet` section is used to specify additional information.
- ❖ Data from one section of the command file is not carried over to the next section. Each section must contain a complete set of data for the command.
- ❖ Service names and parameters are case sensitive.

For example, the following command file executes the `ADD_USER` service and defines attributes for two new users:

```
<?hda version="5.1.1 (build011203)" jcharset=Cp1252 encoding=iso-8859-1?>

# Add users
@Properties LocalData
IdcService=ADD_USER
dName=jsmith
dUserAuthType=Local
dFullName=Jennifer Smith
dPassword=password
dEmail=email@email.com
@end
@ResultSet UserAttribInfo
2
dUserName
AttributeInfo
jsmith
role,contributor,15
@end
<<EOD>>
@Properties LocalData
IdcService=ADD_USER
dName=pwallek
dUserAuthType=Local
dFullName=Peter Wallek
dPassword=password
dEmail=email@email.com
@end
@ResultSet UserAttribInfo
2
dUserName
AttributeInfo
pwallek
```

```

role,contributor,15,account,marketing,7
@end
<<EOD>>

```

## Precedence

---

IdcCommand uses precedence to resolve conflicts among the name/value pairs within the `LocalData` section of the command file. When normal name/value pairs are parsed, they are assumed to be within the `@Properties LocalData` tag. If the section contains HDA tags, the normal name/value pairs take precedence over name/value pairs within the `@Properties LocalData` tag.

For example, if `foo=x` is in a normal name/value pair and `foo=y` is within the `@Properties LocalData` tag, the name/value pair `foo=x` takes precedence because it is outside the tag.

## Special Tags and Characters

---

These special tags and characters can be used in a command file:

Special Character	Description
<code>IdcService=<i>service_name</i></code>	Each section of the command file must specify the name of the service it is calling.
<code>&lt;&lt;EOD&gt;&gt;</code>	The end of data marker. The command file can include one or more sections separated with an end of data marker. See <a href="#">Command File Syntax</a> (page 2-3) for an example.
<code>#</code>	The pound character placed at the beginning of a line indicates that the line is a comment.
<code>\</code>	The backslash is an escape character.

Special Character	Description
@Include <i>filename</i>	This tag enables you to include content from another file at the spot where the @Include tag is placed. This tag can be used to include a complete HDA file or to include shared name/value pairs. This inclusion takes the exact content of the specified file and places it in the location of the @Include tag. A file can be included as many times as desired and an included file may include other files. However, circular inclusions are not allowed.

## CONFIGURATION OPTIONS

To run the IdcCommand utility, specify the following on the command line or in the *intradoc.cfg* configuration file:

Parameter	Required?	Command Line Syntax	Configuration File Syntax
<a href="#">Command File</a> (page 2-5)	Yes	-f name.txt	IdcCommandFile=name.txt
<a href="#">User</a> (page 2-6)	Yes	-u sysadmin	IdcCommandUserName=sysadmin
<a href="#">Log File</a> (page 2-6)	No	-l C:/logs/log.txt	IdcCommandLog=C:/logs/log.txt
<a href="#">Connection Mode</a> (page 2-6)	No	-c auto	ConnectionMode=auto



**Note:** Command line configuration options override the settings in the configuration file.

## Command File

You must specify the name of the command file that contains the service commands and parameters. The command file parameter can specify a full path (such as *C:/command\_files/command.txt*), or it can specify a relative path. See [Command File](#) (page 2-2) for more information.

## User

---

You must specify a content server user name. This user must have permission to execute the services being called.

## Log File

---

You can specify a path and file name for an IdcCommand log file. As each command is executed, a message is sent to the log file, which records the time the command was executed and its success or failure status. If the log file already exists, it will be overwritten with the new message. The log file can be used to display processing information to the user.

- ❖ If the action performed is successful, a “success” message is written to the log file.
- ❖ If the action performed is not successful, an error message is written to the log file.
- ❖ If no log file is specified, information is logged only to the screen.

## Connection Mode

---

You can specify the connection mode for executing the IdcCommand services.

Connection Mode	Description
auto	IdcCommand will attempt to connect to the content server. If this fails, services are executed in stand-alone mode. This is the default connection mode.
server	IdcCommand will execute services only through the content server.
standalone	IdcCommand will execute services in a stand-alone session. There are certain services that cannot be executed in stand-alone mode. In general, these services are performed asynchronously by the server in a background thread. For example, this happens during update or rebuild of the search index.



## RUNNING IDCCommand

---

To run IdcCommand:

1. Create a new IdcCommand working directory.  
Use this directory for your command file and configuration file.
2. Create a [Command File](#) (page 2-2) in the working directory to specify the desired service commands.
3. Copy the *intradoc.cfg* configuration file from `<install_dir>/bin/` into the working directory.



**Important:** Do not delete the *IntradocDir* or *WebBrowserPath* information.

4. Add IdcCommand options to the *intradoc.cfg* file in the working directory. See [Configuration Options](#) (page 2-5) for more information.

```
IdcCommandFile=newfile.hda
IdcCommandUserName=sysadmin
IdcCommandLog=C:/stellent/newlog.txt
```

5. Run the IdcCommand stored in the `<install_dir>/bin` directory:

```
IdcCommand.exe
```

## USING THE LAUNCHER

---

The Launcher is a native C++ application used to manage services in Windows environments and to construct command line arguments and environment settings for the Java VM.

The main operation of the Launcher is to find and read its configuration files, compute any special values, then launch an executable with a command line that it constructs.

Configuration files support Bourne Shell-like substitutions, all of which start with the dollar sign (\$) followed by an alphanumeric identifier or expression inside braces ( { } ).

The Launcher executable is installed in `<install_dir>/shared/os/platform/bin/Launcher`. On UNIX systems, symlinks are created in the `bin/` directory to `Launcher.sh`, a Bourne Shell wrapper which executes the Launcher executable. The purpose of this wrapper is to locate the correct binary Launcher executable for the platform. The term “Launcher” is used here to refer to the native Launcher executable or to the `Launcher.sh` Bourne Shell script.

The Launcher or the symlink to the Launcher.sh must reside in a directory with a valid *intradoc.cfg* configuration file and must have the same name as the Java class file to be launched (case sensitive). The Launcher uses this name to set the environment variable `STARTUP_CLASS`.

On Windows this name is computed by calling `GetModuleFileName()`. On UNIX systems, it is computed by inspecting `argv[0]`. The `PLATFORM` variable is set to the Content Server identifier for the platform. The variable `BIN_DIR` is set to the directory where the Launcher is located.

The Launcher reads a file named *intradoc.cfg* from `BIN_DIR`. This file should contain a value for `IntradocDir`. The `IntradocDir` is used as the base directory for resolving relative paths. Any unqualified path in this document should be taken as relative to the `IntradocDir`. Future releases of the Content Server may change or remove these variable names.

If the *intradoc.cfg* file does not contain a value for `SharedDir`, the Launcher sets `SharedDir` to be `$IntradocDir/shared`. If the Launcher is starting a Windows service, it sets `IS_SERVICE` to 1. If it is unset, the Launcher will also set `PATH_SEPARATOR` to the correct character for the platform.

The Launcher then reads all available configuration files in this order:

1. `$SharedDir/config/resources/launcher.cfg`
2. `$SharedDir/config/resources/launcher-local.cfg`
3. `$BIN_DIR/./config/config.cfg`
4. `$IntradocDir/config/config.cfg`
5. `$IntradocDir/config/config-$PLATFORM.cfg`
6. `$IntradocDir/config/state.cfg`
7. `$SharedDir/os/$PLATFORM/launcher.cfg`
8. `$SharedDir/os/$PLATFORM/launcher-local.cfg`
9. `$BIN_DIR/intradoc.cfg`
10. `$BIN_DIR/intradoc-$PLATFORM.cfg`
11. All files specified on the command line, using the `-cfg` option.

## Quoting

---

The Launcher uses Bourne Shell-like quoting rules. A string can be quoted inside double quotes (“”) to escape spaces. A backslash (\) can precede any character to provide that character. After a final command line is computed, the Launcher separates it into non-quoted spaces. Each string is then unquoted and used as an entry in the `argv` array for the command.

## Computed Settings

---

After reading the configuration files, the Launcher processes variable substitutions. Some variables can have extra computations to validate directories or files, build command line argument lists, or construct PATH-like variables.

These special computations are performed for variables based on their *type*. To set a type for a variable, set `TYPE_variable_name=typename` in any of the configuration files listed previously.

The following list describes Launcher variable types:

### ❖ file

- Examples:

```
TYPE_PASSWD_FILE=file
PASSWD_FILE_sys5=/etc/passwd
PASSWD_FILE_bsd=/etc/master.passwd
```

The type looks for a file. If the value of *variable\_name* is a path to an existing file, it is kept. If not, every variable beginning with *variable\_name\_* is checked. The last value which is a path to an existing file is used for the new value of *variable\_name*.

In this example `PASSWD_FILE` will be set to `/etc/master` if `/etc/master.passwd` exists, or it will be set to `/etc/passwd` if `/etc/passwd` exists. Otherwise, `PASSWD_FILE` will be undefined.

### ❖ directory

- Examples:

```
TYPE_JDK=directory
JDK_java_home=$JAVA_HOME
OS_DIR=$SharedDir/os
DEFAULT_JDK_DIR=$OS_DIR/$PLATFORM
JDK_legacy142=$DEFAULT_JDK_DIR/j2sdk1.4.2_04
JDK_default=$DEFAULT_JDK_DIR/jdk1.5.0_07
```

In this example JDK will be set to the the same value as the last of the JDK\_ variables that is a directory. Typically this would point at the JDK installed with the Content Server. Note that JDK\_java\_home references \$JAVA\_HOME; if a variable isn't defined in any configuration file but is in the environment, the environment value will be used.

#### ❖ executable

- Examples:

```
TYPE_JAVA_EXE=executable
JAVA_EXE_default=java$EXE_SUFFIX
JAVA_EXE_jdk_default=$JDK/bin/java$EXE_SUFFIX
```

The executable type looks for an executable. It works very much like the file type, but will look through every directory in \$PATH for each candidate value. In this example JAVA\_EXE will be set to the java executable in the JDK if it exists. Otherwise it will be set to the first java executable in the PATH.

#### ❖ list

- Examples:

```
TYPE_JAVA_OPTIONS=list
JAVA_MAX_HEAP_SIZE=384
DEFINE_PREFIX=-D
JAVA_OPTIONS_BIN_DIR=${DEFINE_PREFIX}idc.bin.dir=$BIN_DIR
JAVA_OPTIONS_maxheap=${JAVA_MAX_HEAP_SIZE+-Xmx${JAVA_MAX_HEAP_SIZE}m}
JAVA_OPTIONS_service=${IS_SERVICE+$JAVA_SERVICE_EXTRA_OPTIONS}
```

The list type computes a list of options for an executable. Each value that begins with *variable\_name\_* will become a quoted option, and *variable\_name* will be set to the entire list. In this example JAVA\_OPTIONS will be set to the string:

```
"-Didc.bin.dir=/intradocdir/bin/" "-Xmx384m".
```

#### ❖ path

- Examples:

```
TYPE_JAVA_CLASSPATH=path
JAVA_CLASSPATH_legacy=$CLASSPATH
JAVA_CLASSPATH_orig=$IntradocDir/classes
JAVA_CLASSPATH_unpackaged=$SharedDir/classes
JAVA_CLASSPATH_components=$COMPONENTS_CLASSPATH
JAVA_CLASSPATH_server=$SharedDir/classes/server.zip
JAVA_CLASSPATH_refinery=$SharedDir/classes/idcrefinery.zip
JAVA_CLASSPATH_flexion=$SharedDir/classes/flexionxml.jar
JAVA_CLASSPATH_jspserver=$SharedDir/classes/jspserver.jar
JAVA_CLASSPATH_ldap=$SharedDir/classes/ldapjdk.jar
```

The classpath type computes a path-like value. The value of each variable starting with *variable\_name* is appended to the value of *variable\_name* separated by the value of PATH\_SEPARATOR. In this example JAVA\_CLASSPATH will be set to a very long classpath.

#### ❖ lookupstring

- Examples:

```
TYPE_VDK_PLATFORM=lookupstring
PARAMETER_VDK_PLATFORM=${PLATFORM}_${UseVdkLegacySearch+vdk27}
VDK_PLATFORM_aix_vdk27=_rs6k41
VDK_PLATFORM_aix=_rs6k43
VDK_PLATFORM_hpux_vdk27=_hpux11
VDK_PLATFORM_hpux=_hpux11
VDK_PLATFORM_freebsd_vdk27=_ilnx21
VDK_PLATFORM_freebsd=_ilnx21
VDK_PLATFORM_linux_vdk27=_ilnx21
VDK_PLATFORM_linux=_ilnx21
VDK_PLATFORM_solaris_vdk27=_ssol26
VDK_PLATFORM_solaris=_ssol26
VDK_PLATFORM_win32_vdk27=_nti40
VDK_PLATFORM_win32=_nti40
```

The lookupstring uses a second parameter to construct a lookup key for the final value. The second parameter is the value of \$PARAMETER\_variable\_name. If this value is undefined, the current value of variable\_name is used as the lookup key. In this example PARAMETER\_VDK\_PLATFORM has the value of \${PLATFORM}\_ or \${PLATFORM}\_vdk27 depending on the value of UseVdkLegacySearch.

This value is then used to look up the value of the variable VDK\_PLATFORM\_\${PARAMETER\_VDK\_PLATFORM} which is then quoted and assigned to VDK\_PLATFORM.

#### ❖ lookuplist

- Examples:

```
TYPE_STARTUP_CLASS=lookuplist
STARTUP_CLASS_version=Installer --version
STARTUP_CLASS_installer=Installer
STARTUP_CLASS_WebLayoutEditor=IntradocApp WebLayout
STARTUP_CLASS_UserAdmin=IntradocApp UserAdmin
STARTUP_CLASS_RepositoryManager=IntradocApp RepositoryManager
STARTUP_CLASS_Archiver=IntradocApp Archiver
STARTUP_CLASS_WorkflowAdmin=IntradocApp Workflow
STARTUP_CLASS_ConfigurationManager=IntradocApp ConfigMan
```

The `lookuplist` uses a second parameter to construct a lookup key for the final value. The second parameter is the value of `$PARAMETER_variable_name`. If this value is undefined, the current value of `variable_name` is used as the lookup key.

Unlike `lookupstring`, `lookuplist` does not quote the final value. In this example assume the current value of `STARTUP_CLASS` is `version`. `STARTUP_CLASS` will be replaced with the value `Installer --version`.

## Launcher Environment Variables

---

After processing the computed settings, the Launcher iterates over all variables that begin with the string `EXPORT_`. The value of each variable is used as an environment variable name, which has the value of the second half of the `EXPORT_` variable assigned. For example, `EXPORT_IDC_LIBRARY_PATH=LD_LIBRARY_PATH` exports the value of the `IDC_LIBRARY_PATH` variable with the name `LD_LIBRARY_PATH`.

The variable `JAVA_COMMAND_LINE` is used to get the command line. Any command line arguments to the Launcher that haven't been consumed are appended to the command line. On UNIX systems, the command line is parsed and quoting is undone and then `execv` is called. On Windows, a shutdown mutex is created and `CreateProcess` is called with the command line. Care should be taken because `CreateProcess` doesn't undo backslash-quoting.

The principal mechanism for debugging the Launcher is to add the flag `-debug` prior to any arguments for the final command. You can also create a file named `$BIN_DIR/debug.log` which will trigger debug mode and contain the debug output.

The Launcher has knowledge of the following configuration entries which it either sets or used to control its behavior. Note that these configuration variables may change or be removed in future releases of the Content Server:

- ❖ `IDC_SERVICE_NAME`: the name of the win32 service used for service registration, unregistration, startup, and shutdown.
- ❖ `IDC_SERVICE_DISPLAY_NAME`: the display name of the win32 used for service registration.
- ❖ `IntradocDir`: the base directory for relative path names.
- ❖ `IdcBaseDir`: an alternate name for `IntradocDir`.
- ❖ `SharedDir`: set to `$IntradocDir/shared` if otherwise undefined.
- ❖ `SHARED_CONFIG_DIR`: set to `$SharedDir/config` if otherwise unset.
- ❖ `OS_DIR`: set to `$SharedDir/os` if otherwise unset.

- ❖ `PATH_SEPARATOR`: set to either colon (:) or semi-colon (;) if otherwise unset.
- ❖ `STARTUP_CLASS`: set to the name of the Launcher executable.
- ❖ `MUTEX_NAME`: the name used to create a shutdown mutex on win32.
- ❖ `BEFORE_WIN_SERVICE_START_CMD`: if set, is a command line that is executed before a win32 service starts.
- ❖ `UseRedirectedOutput`: if set tells the Launcher on win32 to redirect the output from the Java VM to a file.
- ❖ `ServiceStartupTimeout`: the timeout used for waiting for a Java process to successfully start on win32.



**Tech Tip:** By using the *Launcher.exe*, changing the *status.dat* file, and altering the value of the JVM command line, you could theoretically run any Java program as a Windows service. This is not recommended for normal use, but it does explain the many ways you could configure the Launcher.

## User Interface

---

The UI for the Launcher is exactly the same as the application it launches. For example, if the Launcher is renamed to *IntradocApp*, the following command line arguments are given to launch the Web Layout Editor:

```
IntradocApp WebLayout
```

This will launch the Web Layout Editor as a stand-alone application.

By default, the application will be launched without console output. However, when launching *IdcServer*, *IdcAdmin*, *IdcCommandX*, or the Installer, Java output is printed to the screen. In all other cases, the output is suppressed for a cleaner interface.

For some applications, such as the Batch Loader and the Repository Manager, it is desirable to view the Java output from the application. To force the Launcher to dump the Java output to the screen, use the *-console* flag in this manner:

```
IntradocApp RepMan -console
```

The output will now be written to the console from which the Repository Manager was launched.

If the Launcher is renamed *IdcServer*, *BatchLoader*, *SystemProperties*, or any other Java class that requires no additional parameters, it can be launched with a simple double-click. In other cases, a shortcut can be used to launch them by double-clicking.

## Configuring the Launcher

---

To use the Launcher, you must first rename the *Launcher.exe* file to an executable with the same name as the class file to be launched. Typical examples include *IdcServer.exe* or *IntradocApp.exe*.



**Note:** If you want to make a custom application, you should create the custom directory, and rename the *Launcher.exe* to the service that is to be launched. A valid *intradoc.cfg* file needs to be in the same directory as the executable. The only required parameter is *IntradocDir*; however, other entries can be included to alter the way the Java application is launched.

## Configuration File Example

---

Configuration file example entries:

```
<?cfg jcharset="Cp1252"?>
#Content Server Directory Variables
IntradocDir=C:/stellent/idcml/
CLASSPATH=$COMPUTEDCLASSPATH;$SHAREDDIR/classes/jtds.jar
```

This is sufficient to launch nearly all Content Server applications. Others, such as the Inbound Refinery, require additional classes in the classpath. This file can also be modified to enable Content Server to be run with different Java virtual machines.

The CLASSPATH is designed to look for class files in order of the listed entries. In other words, the Launcher will search the entire *<install\_dir>/classes/* directory before it looks in the *shared* directory or *server.zip* file. This is desirable if the users wish to overload Java classes without patching the Zip file. Additionally, the Launcher can be used to install, uninstall, and run Java applications as Windows Services, provided that they follow the correct API for communicating back to the Launcher. See the source code for *IdcServer.java* or *IdcAdmin.java* for more details on how to make any Java application run as a Windows service with the Launcher.

The COMPUTEDCLASSPATH is used to add class files to the CLASSPATH that the Launcher uses. To add class files, override this flag.



**Note:** The *intradoc.cfg* file is usually altered to include JDBC drivers for their particular database upon install. If you want to use alternate JDBC drivers, place them in the *<install\_dir>/shared/classes* directory, and alter the *intradoc.cfg* file accordingly.

For example, to run Content Server with IBM's virtual machine on a Windows system, the command line would look like:

```
#customized for running IBM's VM
```



```
JAVA_EXE=full path
```

When using a custom JVM, use the full path to the Java executable file to be used.



**Caution:** It is not recommended that you override the JVM command line. If you do, start with the following command line:

```
JvmCommandLine=$JAVA_EXE $JAVA_OPTIONS $JAVA_SERVICE_EXTRA_OPTIONS
$DEFINE_PREFIXjava.endorsed.dirs=$ENDORSEDPATH $APPEND_CLASSPATH
"$CLASSPATH" $STARTUPCLASS
```

If you choose to change which JVM you are using, and if that VM has all the standard Sun SDK jar files, then it is better to use the J2SDK configuration entry to relocate the root directory of the SDK directory rather than use JAVA\_EXE to specify the location of the Java executable (this is not applicable for the IBM VM).

The J2SDK variable changes the directory where the Sun SDK libraries are found (such as tools.jar). If you change this entry without setting the JAVA\_EXE entry then Java executables are assumed to be in the \bin directory of the path in J2SDK. The default value for J2SDK is ...\\shared\\os\\win32\\j2sdk1.4.2\_04.

To override the value for \$JAVA\_OPTIONS, use \$JAVA\_OPTIONS=-server or another similar value.

The following are commonly used command line options. Those options noted with an asterisk (\*) are available on Windows platforms only. Unmarked options are available for Windows or UNIX platforms:

Option	Description
-console	* Forces the Launcher to keep a Windows console window open so that the Java output and error streams are printed to the console.
-debug	Shows paths and variables in use at startup as well as startup errors.
-fileDebug	Similar to the -debug option but this option dumps debug data to the debug.log file. It is usually only set in JAVA_OPTIONS or JAVA_SERVICE_EXTRA_OPTIONS in the intradoc.cfg file to debug Windows services.

Option	Description
-install	* Used to install the Java application referred to by the Launcher as a Windows Service.
-install_autostart	* Similar to the -install option but this option installs the application to start when the server starts.
-uninstall	* Used to uninstall the Java application referred to by the Launcher as a Windows Service.
-remove	* Same as -uninstall.
-dependent <i>service-name</i>	<p>* Makes the Windows service dependent on whether or not the service <i>service-name</i> is also running.</p> <p>This command is useful when you want to make a dependent call for each service.</p> <p>For example, if you want to launch a database before starting the content server, you can specify the content server startup to be dependent on the database startup.</p>
-dependent <i>user password</i>	<p>* Used with -install, installs the service with the credentials of the user specified by <i>user</i> with password <i>password</i>.</p> <p>This command will check the user regardless of the credentials, but may not install the service. The credentials of the user need to extend to the service for the auto-start to run the service automatically.</p> <p>For certain services, such as the Inbound Refinery, the last flag is required so the service can run with higher permissions. The user name must be in the typical Microsoft format DOMAIN\User. Once users change passwords, the service will not be able to log in, and therefore will not run.</p>
-help	Provides verbose output on Launcher use.
-version	Displays the version number for the Launcher and exits.
-asuser <i>user password</i>	* Used during an install to install a service as a specified <i>user</i> with a specific <i>password</i> .

Option	Description
<code>-exec <i>pathname</i></code>	Overrides the <code>argv[0]</code> setting. Used by the <code>Launcher.sh</code> to specify the target <i>pathname</i> because the target of the symlink does not know its source.
<code>-cfg <i>configfilename</i></code>	Specifies additional config files to read before determining computed settings.
<code>-idcServiceName <i>servicename</i></code>	* Specifies the name of the Windows service. This can be used with <code>-remove</code> to uninstall another Content Server service without using that Content Server's Launcher (for example, if an entire installation directory has been removed).



**Tech Tip:** To customize the classpath to alter the system path to load Oracle .dll files, you can change the pathway to:

```
PATH=$SHAREDDIR\os\win32\lib\;$SHAREDDIR\search\vdk\_nti40\bin;$SHAREDDIR\search\vdk\_nti40\filters;%OLDPATH
```

If you want to load custom .dlls, you should put them in the `<install_dir>/shared/os/win32/lib/` directory

## CALLING SERVICES REMOTELY

To use services remotely, you must have these files on the remote machine:

- ❖ `<install_dir>/bin/IdcCommand.exe`
- ❖ `<install_dir>/bin/intradoc.cfg` (same file as on the content server).
- ❖ `<install_dir>/config/config.cfg`

In addition, the following configuration entries must be defined in the `#Additional Variables` section of the `config.cfg` file on the remote machine:

- ❖ `IntradocServerPort=4444`
- ❖ `IntradocServerHostName=IP or DNS`



# IDCCOMMANDX AND IDCCOMMANDUX: ACTIVEX COMMAND UTILITIES

## INTRODUCTION

---

IdcCommandUX is an ActiveX control that allow a program to execute content server services and retrieve file path information. Each control serves as a COM wrapper for the standard IdcCommand services used by content server.

IdcCommandUX is the same as IdcCommandX, except it has been updated to work with multi-byte languages and has more functions which increase its usage for ASPs and SOAP.

This chapter discusses IdcCommandUX, which has more functionality than IdcCommandX.

This chapter covers the following topics:

- ❖ [IdcCommandUX Overview](#) (page 3-2)
- ❖ [Calling IdcCommandUX from an Active Server Page](#) (page 3-4)
- ❖ [Formatting Content with a Resource Include](#) (page 3-9)
- ❖ [Connecting to Content Server from a Remote Machine](#) (page 3-10)
- ❖ [IdcCommandUX Methods](#) (page 3-16)



**Note:** A Visual Basic or Visual C++ development environment is required to use IdcCommandUX.

# IDCCOMMANDUX OVERVIEW

---

This section covers these topics:

- ❖ [Setup](#) (page 3-2)
- ❖ [Calling Procedures](#) (page 3-2)
- ❖ [Executing Services](#) (page 3-3)

## Setup

---

To set up IdcCommandUX, run the IdcCommandUX setup file, which is stored in Extras/IdcCommandUX/Setup.exe on the Content Server DVD.

## Calling Procedures

---

Use one of the following procedures to call IdcCommandUX:

- ❖ [Visual Basic](#) (page 3-2)
- ❖ [Visual C++](#) (page 3-3)

### Visual Basic

To call IdcCommandUX from a Visual Basic environment:

1. Add IdcCommandUX as a control to the Visual Basic project.
2. Create the control as follows:  

```
Set idcCmd=CreateObject("Idc.CommandUX")
```
3. Define and initialize the connection by calling the [init \(deprecated\)](#) (page 3-27) function and defining the *UserName* and *StellentDir* parameters:

```
Dim idcCmd  
idcCmd.init("UserName", "StellentDir")
```

- The *UserName* parameter specifies a user that has permission to execute the services being called by IdcCommandUX.
- The *StellentDir* parameter specifies the complete path to the content server directory that contains the intradoc.cfg configuration file.

For example:

```
Dim idcCmd  
idcCmd.initRemote("sysadmin", "c:\stellent\bin")
```

## Visual C++

Add the IdcCommandUX control to the project and call the desired IdcCommandUX class.

## Executing Services

---

When executing services using IdcCommandUX, keep these points in mind:

- ❖ IdcCommandUX must be initialized with a valid user name and the location of the intradoc.cfg file.
- ❖ Functions that must use HDA format for communication include [computeWebFilePath](#) (page 3-22), [computeNativeFilePath](#) (page 3-18) and [computeURL](#) (page 3-19). For more information on HDA formats, see the *Working with Components* guide.
- ❖ [executeCommand](#) (page 3-24) can take HDA format or SOAP commands. To use SOAP, you must use the [initRemote](#) (page 3-28) function instead of the [init](#) (*deprecated*) function.
- ❖ IdcCommandUX attempts to establish a connection to a running content server. If a connection is not made, it fails.
- ❖ The returned HDA-format string contains information about the success or failure of the command, using the `StatusCode` and `StatusMessage` variables.
  - If the command is successful, `StatusCode` is zero (0), and `StatusMessage` is a login message (“You are logged in as sysadmin”).
  - If the command fails, `StatusCode` is negative (-1), and `StatusMessage` is an error message.



**Note:** See the *Idoc Script Reference Guide* for more information.



**Tech Tip:** See [Using the Launcher](#) (page 2-7) for information on using the Launcher (a native C++ application that allows a Java program to start as a Windows service).

# CALLING IDCCommandUX FROM AN ACTIVE SERVER PAGE

---

Calling IdcCommandUX from an Active Server Page (ASP) consists of the following steps, which are discussed further in this section:

1. [Creating a COM Object](#) (page 3-5)
2. [Initializing the Connection](#) (page 3-5)
3. [Defining the Service and Parameters](#) (page 3-5)
4. [Optional—Formatting Content with a Resource Include](#) (page 3-9)
5. [Formatting Content with a Resource Include](#) (page 3-9)
6. [Retrieving Results](#) (page 3-6)

Two samples are provided: one using HDA and one using SOAP. The examples show the steps needed to obtain a list of content server documents that match a specified criteria. The detailed parts of each step are discussed in the HDA example.

## HDA example

---

```

' Create COM object
Set idcCmd = CreateObject("Idc.CommandUX")
' Initialize the connection to the server
x = idcCmd.initRemote("/stellent/", "socket:localhost:4444", "sysadmin", true)
' Define the service
cmd = "@Properties LocalData" + Chr(10)
cmd = cmd + "IdcService=GET_SEARCH_RESULTS" + Chr(10)
' Define the service parameters
cmd = cmd + "ResultCount=5" + Chr(10)
cmd = cmd + "SortField=dInDate" + Chr(10)
cmd = cmd + "SortOrder=Desc" + Chr(10)
cmd = cmd + "QueryText=dDocType=research" + Chr(10)
' Reference a custom component
cmd = cmd + "MergeInclude=ASP_SearchResults" + Chr(10)
cmd = cmd + "ClassStyle=home-spotlight" + Chr(10)
cmd = cmd + "@end" + Chr(10)
' Execute the command
results = idcCmd.executeCommand(cmd)
' Retrieve results
Response.Write(results)

```



**See Also**

- [Connecting to Content Server from a Remote Machine](#) (page 3-10) for additional information.

**Creating a COM Object**

The first line of code creates the COM object:

```
' Create COM object
Set idcCmd = CreateObject("Idc.CommandUX")
```

**Initializing the Connection**

To initialize the connection to the content server, call the `initRemote` function (see [initRemote](#) (page 3-28) for details about all parameters). In this example:

- ❖ The `HttpWebRoot` parameter specifies a value for the web root as defined in the `config/config.cfg` file.
- ❖ The `idcReference` parameter specifies a string containing information on connection to the content server. This is specified as “socket” followed by the `IntradocServerHostName` and the `IntradocServer Port` address.
- ❖ The `idcUser` is the user you are connecting as.
- ❖ The `isSoap` parameter is a Boolean value indicating if the request is in SOAP XML format or HDA format. In this case it is `FALSE` because it is in HDA format.

```
' Initialize the connection to the server
x = idcCmd.initRemote("/stellent/", "socket:localhost:4444", "sysadmin", false)
```

**Defining the Service and Parameters**

To define the service and parameters, build an HDA-formatted string that contains with the following lines:

```
@Properties LocalData
service
parameters
@end
```



**Important:** The required and optional parameters will vary depending on the service being called. See the *Services Reference Guide* for more information.



**Note:** In this example, the @end string is created after the optional custom component reference. See [Formatting Content with a Resource Include](#) (page 3-9).

## Referencing Custom Resources

You can reference custom resources and pass parameters to a resource include from your ASP as follows:

- ❖ To reference a custom resource include, set the *MergeInclude* parameter to the name of the include.

In this example, the *ASP\_SearchResults* include is used to format the output as HTML rather than a ResultSet. See [Formatting Content with a Resource Include](#) (page 3-9) for more information.

- ❖ To pass a parameter to a resource include, set the variable as name/value pair.

In this example, the *ClassStyle* variable with a value of *home-spotlight* is available to the *ASP\_SearchResults* include.



**Note:** The @end code is required to close the @Properties LocalData section in an HDA-formatted string. See [Defining the Service and Parameters](#) (page 3-5).

```
' Reference a custom component
cmd = cmd + "MergeInclude=ASP_SearchResults" + Chr(10)
cmd = cmd + "ClassStyle=home-spotlight" + Chr(10)
cmd = cmd + "@end" + Chr(10)
```

## Executing the Service

To execute the service, call the [executeCommand](#) (page 3-24) method.



**Note:** After executing the service, you could use the [closeServerConnection](#) (page 3-17) method to make sure that the connection is closed.

```
' Execute the service
results = idcCmd.executeCommand(cmd)
```

## Retrieving Results

The results can either be formatted HTML or a ResultSet.

In this example, the result of the service call is formatted HTML.

```
' Retrieve results  
Response.Write(results)
```

## SOAP Example

---

In this example:

- ❖ The GET\_SEARCH\_RESULTS service is called.
- ❖ The parameters for the service are defined using field/value pairs:
  - The *ResultCount* parameter sets the number of returned results to 5.
  - The *SortField* parameter sorts the returned results by release date.
  - The *SortOrder* parameter orders the returned results in descending order.
  - The *QueryText* parameter defines the query expression as “Content Type matches *research*.”

The [initRemote](#) (page 3-28) function must be used and `isSOAP` must be set to TRUE for a SOAP-formatted request, which is shown in the following example.

```

' Create COM object
Set idcCmd = CreateObject("Idc.CommandUX")
' Initialize the connection to the server
x = idcCmd.initRemote("/stellent/ ", "sysadmin",
    "socket:localhost:4444", true)
' Create the SOAP envelope
cmd = cmd & "<?xml version='1.0' encoding='UTF-8'?" + Chr(10)
cmd = cmd & "<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://
    schemas.xmlsoap.org/soap/envelope/" + Chr(10)
cmd = cmd & "<SOAP-ENV:Body>" + Chr(10)
    ' Define the service
cmd = cmd & "<idc:service xmlns:idc="http://www.stellent.com/
    IdcService/" + Chr(10)
cmd = cmd & "IdcService="GET_SEARCH_RESULTS">" + Chr(10)
    ' Define the service parameters
cmd = cmd & "<idc:document>" + Chr(10)
cmd = cmd & "<idc:field name="NoHttpHeaders">1</idc:field>" +
    Chr(10)
cmd = cmd & "<idc:field name="ClientEncoding">UTF8</idc:field>"
    + Chr(10)
cmd = cmd & "<idc:field name="QueryText">dDocType
    &lt;matches> research</idc:field>" + Chr(10)
cmd = cmd & "<idc:field name="ResultCount">5</idc:field>" +
    Chr(10)
cmd = cmd & "<idc:field name="SortOrder">Desc</idc:field>" +
    Chr(10)
cmd = cmd & "<idc:field name="SortField">dInDate</idc:field>" +
    Chr(10)
cmd = cmd & "</idc:document>" + Chr(10)
cmd = cmd & "</idc:service>" + Chr(10)
cmd = cmd & "</SOAP-ENV:Body>" + Chr(10)
cmd = cmd & "</SOAP-ENV:Envelope>" + Chr(10)
' End SOAP envelope and execute the command
results= idcCmd.executeCommand(cmd)
' Retrieve results
Response.Write(results)

```

## FORMATTING CONTENT WITH A RESOURCE INCLUDE

This section provides an example of a custom resource include that is used to format the output of a service executed by IdcCommandUX.

In the example described in [Calling IdcCommandUX from an Active Server Page](#) (page 3-4), the *ASP\_SearchResults* resource include is used to format the output of a search function and return HTML rather than a ResultSet:

```
<@dynamichtml ASP_SearchResults@>
<table border=0>
  <$loop SearchResults$>
  <tr class="site-default">
    <td class="<$ClassStyle$">
      <a href="<$URL$>" target=new><$dDocTitle$></a><br>
      <$xAbstract$>
    </td>
  </tr>
  <$endloop$>
</table>
<@end@>
```

- ❖ The `<@dynamichtml ASP_SearchResults@>` entry defines the name of the resource include. The `<@end@>` entry ends the resource definition.
- ❖ The code defined between the `<$loop SearchResults$>` and `<$endloop$>` entries will be executed for each content item in the SearchResults ResultSet, which includes all documents that matched the query defined for the GET\_SEARCH\_RESULTS service.
- ❖ The `<td class="<$ClassStyle$">` entry displays the value of the `<$ClassStyle$>` Idoc Script variable. In this example, the `ClassStyle` value was passed in on the API call.
- ❖ The `<a href="<$URL$>" target=new><$dDocTitle$></a>` entry displays the Title of the current content item as a link to the file.
- ❖ The `<$xAbstract$>` entry displays the Abstract value for the current content item.

The HTML generated and returned to the Active Server Page from this resource include would have this format:

```
<table border=0>
<tr class="site-default">
<td class="home-spotlight">
<a href="/stellent/dir/dir/xyz.htm" target=new>Article 1</a><br>
This is the abstract for Article 1
</td>
<td class="home-spotlight">
<a href="/stellent/dir/dir/xyz.htm" target=new>Article 2</a><br>
This is the abstract for Article 2
</td>
<td class="home-spotlight">
<a href="/stellent/dir/dir/xyz.htm" target=new>Article 3</a><br>
This is the abstract for Article 3
</td>
<td class="home-spotlight">
<a href="/stellent/dir/dir/xyz.htm" target=new>Article 4</a><br>
This is the abstract for Article 4
</td>
<td class="home-spotlight">
<a href="/stellent/dir/dir/xyz.htm" target=new>Article 5</a><br>
This is the abstract for Article 5
</td>
</tr>
</table>
```

Displaying this HTML page in a browser would look like this:

<a href="#">Article 1</a> This is the abstract for Article 1	<a href="#">Article 2</a> This is the abstract for Article 2	<a href="#">Article 3</a> This is the abstract for Article 3	<a href="#">Article 4</a> This is the abstract for Article 4	<a href="#">Article 5</a> This is the abstract for Article 5
--	--	--	--	--

## CONNECTING TO CONTENT SERVER FROM A REMOTE MACHINE

---

This section describes how to establish a connection to the content server from a remote machine using IdcCommandUX from an Active Server Page. These steps are required :

1. [Creating Variables](#) (page 3-12)
2. [Creating a COM Object](#) (page 3-13)
3. [Initializing the Connection](#) (page 3-13)

4. [Returning Connection Status](#) (page 3-14)
5. [Defining the Service and Parameters](#) (page 3-14)
6. [Executing the Service](#) (page 3-16)
7. [Retrieving Results](#) (page 3-16)

The example in this section calls the CHECKIN\_UNIVERSAL service to provide a checkin function from a remote machine.

## Coding the ASP Page

---

This section provides the steps for coding an Active Server Page to access the content server from a remote machine.



**Important:** This code does not check for an error condition.

```

' Create variables
Dim idccommand, sConnect, str
' Create COM object
Set idccommand = Server.CreateObject("idc.CommandUX")
' Initialize the connection to the server
x = idccommand.initRemote ("/stellent/ ", "sysadmin", "socket:localhost:4444",
false)
' Return connection status (optional)
sConnect = idccommand.connectToServer
if sConnect then
Response.Write "Connected"
else
Response.Write "Not Connected"
end if
str = "@Properties LocalData" & vbcrLf
' Define the service
str = str + "IdcService=" & "CHECKIN_UNIVERSAL" & vbcrLf
' Define the service parameters
str = str + "doFileCopy=1" & vbcrLf
str = str + "dDocName=RemoteTestCheckin23" & vbcrLf
str = str + "dDocTitle=Test1" & vbcrLf
str = str + "dDocType=ADACCT" & vbcrLf
str = str + "dSecurityGroup=Public" & vbcrLf
str = str + "dDocAuthor=sysadmin" & vbcrLf
str = str + "dDocAccount=" & vbcrLf
str = str + "primaryFile:path=C:/inetpub/Scripts/query2.asp" & vbcrLf
str = str + "@end" & vbcrLf
' Execute the command
res=idccommand.executeCommand(str)
' Return connection status
sClosed = idcCmd.closeServerConnection
if sClosed then
Response.Write "Server connection closed"
else
Response.Write "Failed to close server connection"
end if
' Retrieve results
Response.Write(res)

```

## Creating Variables

The following variables need to be created for this example:



- ❖ **idccommand**—The name of the COM object.
- ❖ **sConnect**—The status of the connection to the content server.
- ❖ **str**—The HDA-formatted string that defines the service and its parameters.

```
' Create variables  
Dim idccommand, sConnect, str
```

## Creating a COM Object

The first line of code creates the COM object:

```
' Create COM object  
Set idccommand = Server.CreateObject("idc.CommandUX")
```

## Initializing the Connection

To initialize the connection to the content server:

```
' Initialize the connection to the server  
x = idccommand.initRemote ("/stellent/ ", "sysadmin", "socket:localhost:4444",  
false)
```

## Returning Connection Status

In this example, the [connectToServer](#) (page 3-23) and [closeServerConnection](#) (page 3-17) methods are used to return connection status information before and after the service is executed.

```
' Return connection status
sConnect = idccommand.connectToServer
if sConnect then
Response.Write "Connected"
else
Response.Write "Not Connected"
end if
...
' Return connection status
sClosed = idcCmd.closeServerConnection
if sClosed then
Response.Write "Server connection closed"
else
Response.Write "Failed to close server connection"
end if
```

## Defining the Service and Parameters

To define the service and parameters, build an HDA-formatted string that contains the following lines:

```
@Properties LocalData
service
parameters
@end
```



**Important:** The required and optional parameters will vary depending on the service being called. See the *Services Reference Guide* for more information.

In this example:

- ❖ The CHECKIN\_UNIVERSAL service is called.
- ❖ The parameters for the service are defined using field/value pairs:
  - The *doFileCopy* parameter is set to TRUE (1), so the file will not be deleted from hard drive after successful check in.
  - The *dDocName* parameter defines the Content ID.
  - The *dDocTitle* parameter defines the Title.
  - The *dDocType* parameter defines the Type.

- The *dSecurityGroup* parameter defines the Security Group.
- The *dDocAuthor* parameter defines the Author.
- The *dDocAccount* parameter defines the security account. (If accounts are enabled, this parameter is required.)
- The *primaryFile* parameter defines original name for the file and the absolute path to the location of the file as seen from the server.



**Important:** The required parameters will vary depending on the service called. See the *Services Reference Guide* for additional information.

```
str = "@Properties LocalData" & vbcrLf
' Define the service
str = str + "IdcService=" & "CHECKIN_UNIVERSAL" & vbcrLf
' Define the service parameters
str = str + "doFileCopy=1" & vbcrLf
str = str + "dDocName=RemoteTestCheckin23" & vbcrLf
str = str + "dDocTitle=Test1" & vbcrLf
str = str + "dDocType=ADACCT" & vbcrLf
str = str + "dSecurityGroup=Public" & vbcrLf
str = str + "dDocAuthor=sysadmin" & vbcrLf
str = str + "dDocAccount=" & vbcrLf
str = str + "primaryFile:path=C:/inetpub/Scripts/query2.asp" & vbcrLf
str = str + "@end" & vbcrLf
```

## Executing the Service

To execute the service, call the [executeCommand](#) (page 3-24) method.

```
' Execute the service  
res=idccommand.executeCommand(str)
```

## Retrieving Results

In this example, the result of the CHECKIN\_UNIVERSAL service call is formatted HTML.

```
' Retrieve results  
Response.Write(res)
```

# IDCCOMMANDUX METHODS

---

This section describes the following IdcCommandUX methods:

- ❖ [addExtraHeadersForCommand](#) (page 3-17)
- ❖ [closeServerConnection](#) (page 3-17)
- ❖ [computeNativeFilePath](#) (page 3-18)
- ❖ [computeURL](#) (page 3-19)
- ❖ [computeWebFilePath](#) (page 3-22)
- ❖ [connectToServer](#) (page 3-23)
- ❖ [executeCommand](#) (page 3-24)
- ❖ [executeFileCommand](#) (page 3-25)
- ❖ [forwardRequest](#) (page 3-26)
- ❖ [getLastErrorMessage](#) (page 3-26)
- ❖ [initRemote](#) (page 3-28)
- ❖ [init \(deprecated\)](#) (page 3-27)
- ❖ [initRemote](#) (page 3-28)



**Important:** All parameters are required unless otherwise indicated.

## addExtraHeadersForCommand

---

This command adds extra HTTP-like headers to a command.

- ❖ For security reasons, some parameters can only be passed in the headers.
- ❖ The most common use for this command is to set the values for `EXTERNAL_ROLES` and `EXTERNAL_ACCOUNTS` in a request.
- ❖ Values must be all on one string and separated by a carriage return and a line feed.

### **Example**

The following is an ASP example:

```
extraHeaders = "EXTERNAL_ROLES=contributor" _  
              + vbCrLf _  
              + "EXTERNAL_ACCOUNTS=my_account"  
idcCmd.addExtraHeadersForCommand(extraHeaders)
```

## closeServerConnection

---

```
Public Sub closeServerConnection()
```

### **Description**

Closes the server connection.

- ❖ This method does not have to be called, because the [executeCommand](#) (page 3-24) method automatically closes a connection after executing a service. It is provided only as a convenience for managing the state of the connection.

### **Parameters**

None

### **Output**

- ❖ Returns `TRUE` if the connection is closed.
- ❖ Returns `FALSE` if the connection failed to close.

### **Example**

This ASP example passes the result of the *closeServerConnection* method to a variable and uses an if/else statement to return a connection status message:

```
sClosed = idcCmd.closeServerConnection
if sClosed then
Response.Write "Server connection closed"
else
Response.Write "Failed to close server connection"
end if
```

### **See Also**

- [executeCommand](#) (page 3-24)
- [connectToServer](#) (page 3-23)

## **computeNativeFilePath**

---

Public Function computeNativeFilePath(Data As String) As String

### **Description**

#### **HDA-only function.**

Returns the path of a native file as a string.

- ❖ This function is generally used for processing native files to perform actions such as bulk file loading or retrieval.
- ❖ To determine the values for the required parameters (such as *dDocType* and *dID*), you can reference the ResultSet returned from a DOC\_INFO or SEARCH\_RESULTS service call.
  - The DOC\_INFO service can be used to specify previous revisions (DOC\_INFO returns a list of previous revision labels).
  - The SEARCH\_RESULTS service returns only enough data to specify the most recent revision of a content item.

### **Parameters**

- ❖ Data: An HDA-formatted string that defines the content item:
  - **dDocType**—The content item Type, such as *ADACCT* or *FILES*.
  - **dID**—The generated content item revision ID.
  - **dExtension**—The file extension such as *HCSF*, *DOC*, or *TXT*.

- **dDocAccount**—The account for the content item. If accounts are enabled, this parameter must be defined.



**Note:** Do not confuse the Content ID (*dDocName*) with the internal content item revision identifier (*dID*). The *dID* is a generated reference to a specific revision of a content item.

## Output

- ❖ Returns a string that defines *NativeFilePath* as the value of the string passed in as a parameter. For example:  
NativeFilePath=c:\stellent\vault\adacct\1.doc
- ❖ Returns an HDA string containing *StatusCode* and *StatusMessage*.
  - If the command is successful, *StatusCode* is zero (0), and *StatusMessage* is a login message (“You are logged in as sysadmin”).
  - If the command fails, *StatusCode* is negative (-1), and *StatusMessage* is an error message.
  - Returns FALSE if there is a connection failure.

## Example

This is an example of an HDA-formatted string:

```
String str = "@Properties LocalData\n"+
"dDocType=ADACCT\n"+
"dID=67\n"+
"dExtension=DOC\n"+
"dDocAccount=mainaccount\n"+
"@end\n";
```

## computeURL

---

```
Public Function computeURL(Data As String, IsAbsolute As Boolean) As String
```

### Description

#### HDA-only function.

Returns the URL of a content item as a string.

- ❖ A relative or absolute URL can be supplied to the content server.
  - When a relative URL is defined, the function evaluates the URL as a location valid on the local server.

For example:

```
/stellent/groups/Public/documents/FILE/doc.txt
```

- When an absolute URL is defined, the function returns the absolute URL path.

For example:

```
http://server/stellent/groups/Public/documents/FILE/doc.txt
```

- ❖ To determine the values for the content server parameters (*HttpRelativeWebRoot* and *HttpServerAddress*), you can reference the properties data returned from a GET\_DOC\_CONFIG\_INFO service call.

- ❖ To determine the values for the required content item parameters (such as *dSecurityGroup* and *dDocType*), you can reference the ResultSet returned from a DOC\_INFO or SEARCH\_RESULTS service call.

- The DOC\_INFO service can be used to specify previous revisions (DOC\_INFO returns a list of previous revision labels).
- The SEARCH\_RESULTS service returns only enough data to specify the most recent revision of a content item.

- ❖ To return the URL for a specific revision and rendition, use the content item revision label (*dRevLabel*) and the file extension (*dWebExtension*) entries. For example:

```
dDocName=test10  
dRevLabel=2  
dWebExtension=pdf
```

- ❖ To return the URL for the most recent revision, the content item revision label (*dRevLabel*) entry can be omitted. For example, defining just the Content ID (*dDocName*) and the file extension (*dWebExtension*) returns the most recent revision:

```
dDocName=test11  
dWebExtension=html
```

## Parameters

- ❖ Data: An HDA-formatted string that defines the content item:
  - **HttpRelativeWebRoot**—The web root directory as a relative path, such as */stellent/*. This entry is required for a relative URL, and is optional for an absolute URL.
  - **HttpServerAddress**—The domain name of the content server, such as *testserver17* or *mycomputer.com*. (The server address is specified as a partial URL such as *mycomputer.com* rather than a full address such as *http://www.mycomputer.com/*). This entry is required for an absolute URL, and is optional for a relative URL.



- **dSecurityGroup**—The security group, such as *Public* or *Secure*.
  - **dDocType**—The Type, such as *ADACCT* or *FILES*.
  - **dDocName**—The Content ID, such as *test10* or *hr\_0005467*.
  - **dWebExtension**—The file extension of the web-viewable file, such as *xml*, *html*, or *txt*.
  - **dDocAccount**—The account for the content item. If accounts are enabled, this parameter must be defined.
  - **dRevLabel** (optional)—The revision label for the content item. If defined, the specific revision will be referenced.
- ❖ **IsAbsolute**: Set to TRUE (1) to define an absolute URL address.



**Note:** Do not confuse the Content ID (*dDocName*) with the internal content item revision identifier (*dID*). The *dID* is a generated reference to a specific revision of a content item.

## Output

- ❖ Returns a string that defines *URL* as the value of the string passed in as a parameter.  
For example:  
URL=http://server/stellent/groups/public/documents/FILE/doc.txt
- ❖ Returns an HDA string containing *StatusCode* and *StatusMessage*.
  - If the command is successful, *StatusCode* is zero (0), and *StatusMessage* is a login message (“You are logged in as sysadmin”).
  - If the command fails, *StatusCode* is negative (-1), and *StatusMessage* is an error message.
  - Returns FALSE if there is a connection failure.

## Example

This is an example of an HDA-formatted string:

```
String str = "@Properties LocalData\n"+
  "HttpServerAddress=testserver17\n"+
  "HttpRelativeWebRoot=/stellent/\n"+
  "dDocAccount=mainaccount\n"+
  "dSecurityGroup=Public\n"+
  "dDocType=ADACCT\n"+
  "dDocName=test11\n"+
  "dWebExtension=html\n"+
  "@end\n";
```

## computeWebFilePath

---

Public Function computeWebFilePath(Data As String) As String

### Description

#### HDA-only function.

Returns the path of a web-viewable file as a string.

- ❖ This function is generally used for processing web-viewable text files (such as XML) to perform actions such as bulk file loading or retrieval.
- ❖ Using *computeWebFilePath* instead of [computeNativeFilePath](#) (page 3-18) provides the advantage of needing only the Content ID (*dDocName*) rather than the specific revision ID (*dID*) to return the most recent revision.
- ❖ To determine the values for the required parameters (such as *dSecurityGroup* and *dDocType*), you can reference the ResultSet returned from a DOC\_INFO or SEARCH\_RESULTS service call.
  - The DOC\_INFO service can be used to specify previous revisions (DOC\_INFO returns a list of previous revision labels).
  - The SEARCH\_RESULTS service returns only enough data to specify the most recent revision of a content item.

### Parameters

- ❖ Data: An HDA-formatted string that defines the content item:
  - **dSecurityGroup**—The security group, such as *Public* or *Secure*.
  - **dDocType**—The content item Type, such as *ADACCT* or *FILES*.
  - **dDocName**—The Content ID, such as *test10* or *hr\_0005467*.
  - **dWebExtension**—The file extension of the web-viewable file, such as *xml*, *html*, or *txt*.
  - **dDocAccount**—The account for the content item. If accounts are enabled, this parameter must be defined.



**Note:** Do not confuse the Content ID (*dDocName*) with the internal content item revision identifier (*dID*). The *dID* is a generated reference to a specific revision of a content item.

## Output

- ❖ Returns a string that defines *WebFilePath* as the value of the string passed in as a parameter. For example:

```
WebFilePath=http:\\testserver17.stellent.com\\stellent\\groups\\main\\documents\\test.xml
```

- ❖ Returns an HDA string containing *StatusCode* and *StatusMessage*.
  - If the command is successful, *StatusCode* is zero (0), and *StatusMessage* is a login message (“You are logged in as sysadmin”).
  - If the command fails, *StatusCode* is negative (-1), and *StatusMessage* is an error message.
  - Returns FALSE if there is a connection failure.

## Example

This is an example of an HDA-formatted string:

```
String str = "@Properties LocalData\n"+
"dDocAccount=mainaccount\n"+
"dSecurityGroup=Public\n"+
"dDocType=ADACCT\n"+
"dDocName=test11\n"+
"dWebExtension=xml\n"+
"@end\n";
```

## connectToServer

---

```
Public Function connectToServer() As Boolean
```

### Description

Establishes a connection to the server.

- ❖ The connection is held open until a command is executed. After a command is executed, the connection is closed automatically.
- ❖ This method does not have to be called, because the [executeCommand](#) (page 3-24) method automatically opens a connection to execute a service. It is provided only as a convenience for managing the state of the connection.

### Parameters

None

## **Output**

- ❖ Returns TRUE if the connection is opened.
- ❖ Returns FALSE if there is a connection failure.

## **Example**

This ASP example passes the result of the *connectToServer* method to a variable and uses an if/else statement to return a connection status message:

```
sConnect = idcCmd.connectToServer
if sConnect then
Response.Write "Connected"
else
Response.Write "Not Connected"
end if
```

## **See Also**

- [executeCommand](#) (page 3-24)
- [closeServerConnection](#) (page 3-17)

# **executeCommand**

---

```
Public Sub executeCommand(Data As String)
```

## **Description**

Executes a content server service.

- ❖ This method evaluates whether a connection has already been established with a *connectToServer* call. If a connection exists, it will use the open connection. If a connection does not exist, it will establish a connection.
- ❖ On completion of the command, the connection will be closed.

## **Parameters**

- ❖ **Data**: An HDA-formatted string that defines the *IdcService* command and any service parameters. For example:

```
@Properties LocalData
IdcService=GET_SEARCH_RESULTS
ResultCount=5
SortField=dInDate
SortOrder=Desc
```

```
QueryText=dDocType=research
@end@
```

This can also be a SOAP-formatted message as shown in the previous example ([SOAP Example](#) (page 3-7)). See also [initRemote](#) (page 3-28).

## Output

- ❖ Returns a string representing an HDA file that holds the original request as well as the results.
- ❖ Returns an HDA string containing `StatusCode` and `StatusMessage`.
  - If the command is successful, `StatusCode` is zero (0), and `StatusMessage` is a login message (“You are logged in as sysadmin”).
  - If the command fails, `StatusCode` is negative (-1), and `StatusMessage` is an error message.
  - Returns FALSE if there is a connection failure.
- ❖ The return string is SOAP-formatted XML if a SOAP request was sent.

## Example

This ASP example executes the command specified in the data string defined by the `cmd` variable:

```
results = idcCmd.executeCommand(cmd)
```

## See Also

- [connectToServer](#) (page 3-23)
- [closeServerConnection](#) (page 3-17)

## executeFileCommand

---

```
executeFileCommand (requestString)
```

### Description

This function is used to execute a service request, then pipe the raw response to the client. This command is identical to `executeCommand` but can only be called on an Active Server Page (ASP).

- ❖ The response from the content server is redirected back to the client's browser (this is different from the response via `executeCommand`, in which the response is given as a string which can then be manipulated on the ASP).
- ❖ This is useful for `GET_FILE` and similar services in which you need to transfer binary files from the content server to a client browser through an ASP.
- ❖ This function returns extra headers unless the request parameters are passed as environment variables.
- ❖ *requestString* is the name of the service request.
- ❖ See [executeCommand](#) (page 3-24) for more information.

### **Parameters**

None

## **forwardRequest**

---

`forwardRequest()`

### **Description**

This function is used to forward a multipart form post to the content server. This is useful for executing checkins.

### **Parameters**

None

## **getLastErrorMessage**

---

`getLastErrorMessage()`

### **Description**

This method retrieves the specific error details for a communication or configuration error. For example, if you do not put in the correct hostname for making a connection, this method returns the connection error. It does not return a value if the error is returned by the Content Server as part of the return value for a request.

## Parameters

None

## Example

This example creates an object and initializes a connection to the server.

```
Set idcCmd = Server.CreateObject("Idc.CommandUX")

x = idcCmd.init("sysadmin", "c:\stellent\bin")
If x = false Then
y = idcCmd.getLastErrorMessage()
Response.Write(y)
End If
```

## init (deprecated)

---

```
Public Function init(Username As String, StellentDir As String) As Boolean
```

### Description

This is a deprecated function. Use [initRemote](#) (page 3-28).

This function initializes the connection to the content server.

- ❖ This function must be called before establishing a connection to the content server or executing a content server service.

### Required Parameters

- ❖ **UserName:** A valid content server user that has permission to execute the services specified in the IdcCommandUX call.
- ❖ **StellentDir:** The complete path to the directory that contains the *intradoc.cfg* configuration file. If this parameter is not set, the current working directory is used.

### Output

Returns FALSE if the service fails to execute.

## Example

This ASP example initializes the connection to the server:

```
❖ x = idcCmd.init("sysadmin", "c:/stellent_english/bin/")
```

## initRemote

---

```
initRemote(HttpWebRoot, idcReference, idcUser, isSoap)
```

### **Description**

This function initializes the module to connect to a content server. Note that you must first declare `idcCmd`, as in this example:

```
Dim idcCmd  
idcCmd.initRemote("stellent", "socket:test204:4444", "sysadmin", "false")
```

### **Required Parameters**

- ❖ `HttpWebRoot`: The IdocScript value for `HttpWebRoot`.
- ❖ `idcReference`: A string containing information on how to connect to the content server in the form `socket:hostname:port`. This is typically `socket:localhost:4444`. The `hostname` should be identical to `IntradocServerHostName` and `port` identical to `IntradocServerPort`.
- ❖ `idcUser`: The user you are connecting as.
- ❖ `isSoap`: A Boolean value indicating if the request is in SOAP XML format or HDA format.



# IDCCLIENT OCX COMPONENT

## INTRODUCTION

---

An Object Linking and Embedding Control Extension (OCX) control is provided for connecting to a remote content server and executing Content Server services. The IdcClient OCX control is used within a Windows Visual Basic development environment to gain access to the content and content management functions within Content Server.

This section provides a description of the IdcClient OCX control, setup instructions, and lists the events, methods, and properties. The *IdcClient.ocx* control is used to connect to a remote content server and perform typical server functions.

This chapter covers the following topics:

- ❖ [IdcClient OCX Description](#) (page 4-2)
- ❖ [IdcClient OCX Control Setup](#) (page 4-5)
- ❖ [IdcClient Events](#) (page 4-16)
- ❖ [IdcClient Methods](#) (page 4-18)
- ❖ [IdcClient Properties](#) (page 4-34)



**Note:** A Visual Basic or Visual C++ development environment is required to use the IdcClient OCX component.

## IDCCCLIENT OCX DESCRIPTION

---

This section provides a general description of the IdcClient OCX control and basic information on events, methods, and properties. The IdcClient OCX interface is also discussed.

This section covers the following topics:

- ❖ [General Description](#) (page 4-2)
- ❖ [Events, Methods, and Properties](#) (page 4-3)
- ❖ [IdcClient OCX Interface](#) (page 4-4)

### General Description

---

IdcClient is an ActiveX control that allows a program to perform actions such as executing a service and retrieving file path information. The IdcClient control is also a wrapper for the Microsoft Internet Explorer browser.

The IdcClient OCX control is designed to use the Unicode standard and in most cases exchanges data with the content server in UTF-8 format. Unicode uses two bytes (16 bits) of storage per character and can represent characters used in a wide range of languages (e.g., English, Japanese, Arabic). Since English language ASCII (American Standard Code for Information Interchange) characters only require one byte (8 bits), when an ASCII character is represented the upper byte of each Unicode character is zero.



**Important:** IdcClient OCX is built atop the Microsoft Layer for Unicode, which allows Unicode applications to run on Win9x platforms. When distributing the IdcClient OCX Control on 9x platforms, the "unicows.dll" needs to also be distributed. This companion DLL cannot be distributed on Windows-based systems.



**Note:** Refer to the Unicode Consortium on the Web for additional information on the Unicode standard (<http://www.unicode.org/>).

In most cases, the methods use the serialized HDA format for communication. A serialized HDA format is a Java method used for communication. The returned serialized HDA format string contains information about the success or failure of the command.

The IdcClient OCX control provides functionality that can be performed with a method call. Methods perform actions and often return results. Information is passed to methods using parameters. Some functions do not take parameters; some functions take one

parameter; some take several. For example, a function with two parameters passed as strings would use this format:

```
Function(Parameter As String, Parameter As String) As String
```



**Note:** See the *Services Reference Guide* for additional information.

- ❖ IdcClient OCX enables users to write client applications to execute services. The OCX control takes name/value pairs containing commands and parameters and calls the specified services. Execution results are passed back to the calling program.
- ❖ IdcClient OCX requires a username and password to execute the commands. The user must have the appropriate permissions to execute the commands. Some commands will require an administrative access level, other commands may require only write permission.

## Events, Methods, and Properties

---

The IdcClient OCX control is used to connect to a remote Content Server and perform server functions. This section provides a basic overview on Visual Basic events, methods, and properties.

### OCX Events

Events are executed when the user or server performs an action.

For example:

- ❖ The `IntradocBrowserPost` event executes every time a user submits a form from within a browser.
- ❖ The `IntradocServerResponse` event executes after the server completes a requested action.



**Note:** See [IdcClient Events](#) (page 4-16) for additional information.

### OCX Methods

The Visual Basic Standard Controls provide methods that are common to every Visual Basic development environment. In addition, the IdcClient OCX control provides methods

that are private and unique to this specific control. These methods are used to perform or initiate an action rather than setting a characteristic.

For example:

- ❖ The `AboutBox()` method launches the About box containing product version information.
- ❖ The `GoCheckinPage` method checks in a new content item or a content item revision.



**Note:** See [IdcClient Methods](#) (page 4-18) for additional information.

## OCX Properties

Properties describe or format an object and can be modified with code or by using the property window in the Visual Basic development environment. Properties describe the basic characteristic of an object.

For example:

- ❖ The `UserName` property provides the assigned user name.
- ❖ The `WorkingDir` property specifies the location where downloaded files are placed.



**Note:** See [IdcClient Properties](#) (page 4-34) for additional information.

## IdcClient OCX Interface

---

The IdcClient OCX control is used within a Windows Visual Basic development environment to gain access to the content and content management functions within Content Server. The OCX integration is designed to call services in a visual development environment, or to connect to a remote content server.

In most cases, methods use the serialized HDA format for communication. The returned serialized HDA format string contains information about the success or failure of the command. The `StatusCode` will be negative if a failure occurs, and `StatusMessage` will indicate the error. If the returned HDA does not contain a `StatusCode` parameter, the service call succeeded.

# IDCCCLIENT OCX CONTROL SETUP

---

This section provides a the steps required to setup the IdcClient OCX component and also provides information on creating a visual interface in the the Microsoft Visual Basic development environment.

This section covers the following topics:

- ❖ [Component Setup](#) (page 4-5)
- ❖ [Creating a Visual Interface](#) (page 4-5)

## Component Setup

---

Follow these steps to set up the IdcClient OCX component in the Microsoft Visual Basic development environment:

1. Create a new project.
2. Select **Project—Components**.
3. Browse to the IdcClient.ocx file on your system and click **Open**.

The IdcClient module is added to the Component Controls list.

4. Ensure that the check box for *IdcClient ActiveX Control* module is enabled and click **OK**.

The IdcClient OCX control is placed in the list of controls.

5. Optional—You can use the Visual Basic development environment to build your own visual interface or follow the steps provided in [Creating a Visual Interface](#) (page 4-5) to build a basic visual interface.

## Creating a Visual Interface

---

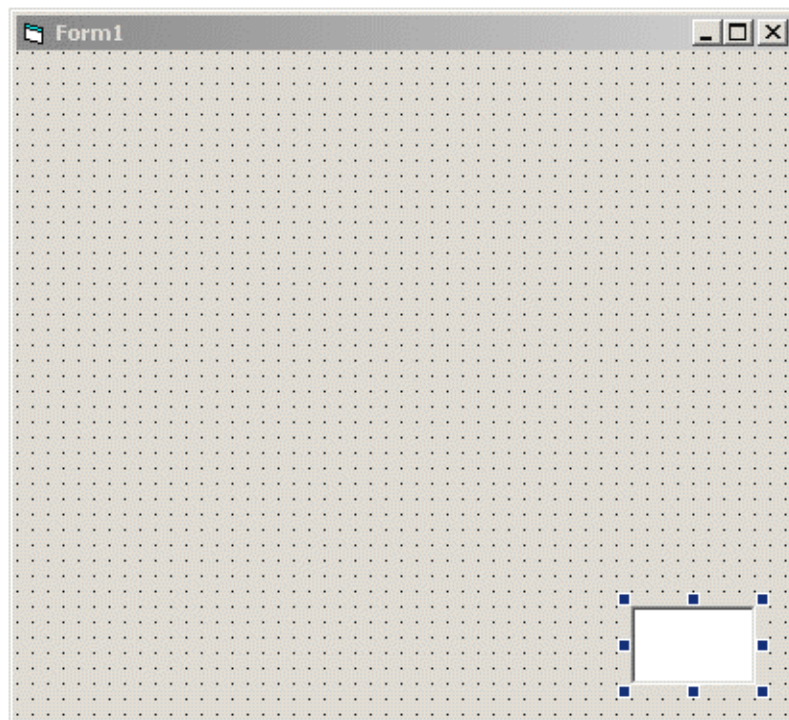


**Note:** It is assumed that a Visual Basic project has been created and the IdcClient OCX control has been placed in the list of controls. See [Component Setup](#) (page 4-5) for additional information.

Follow these steps to build a basic visual interface:

1. Select the control and draw it on the Visual Basic form (see Figure 4-1).

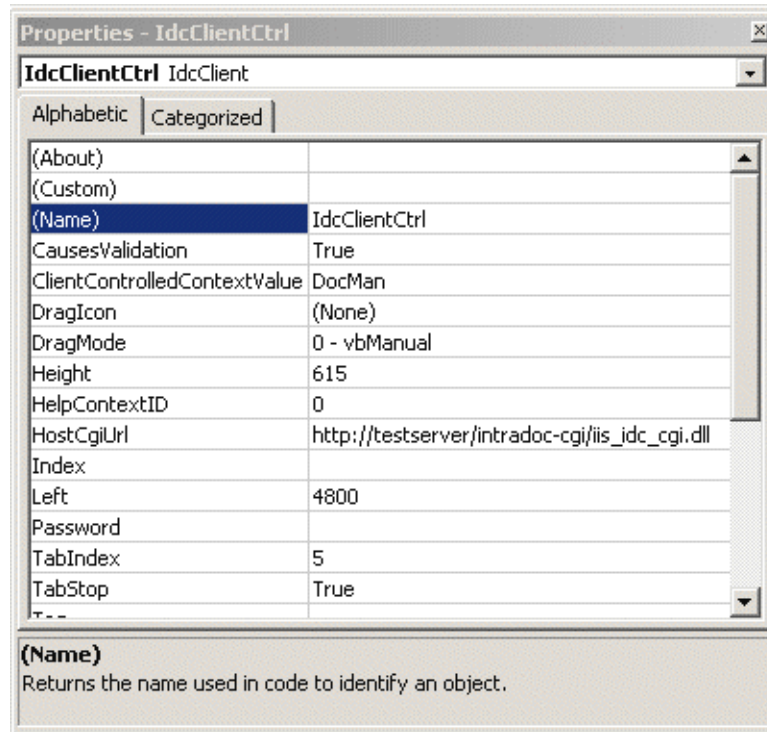
**Figure 4-1** OCX control drawn on a Visual Basic form



2. From the drop-down list of the Properties Window, select the IdcClient OCX control (if the Properties Window is not currently displayed select **View—Properties Window** from the main menu).
3. Rename the IdcClient OCX control IdcClientCtrl.
4. Define the HostCgiUrl to reference the iss\_idc\_cgi.dll for your particular instance (see Figure 4-2).

For example:

```
http://testserver/intradoc-cgi/iss_idc_cgi.dll
```

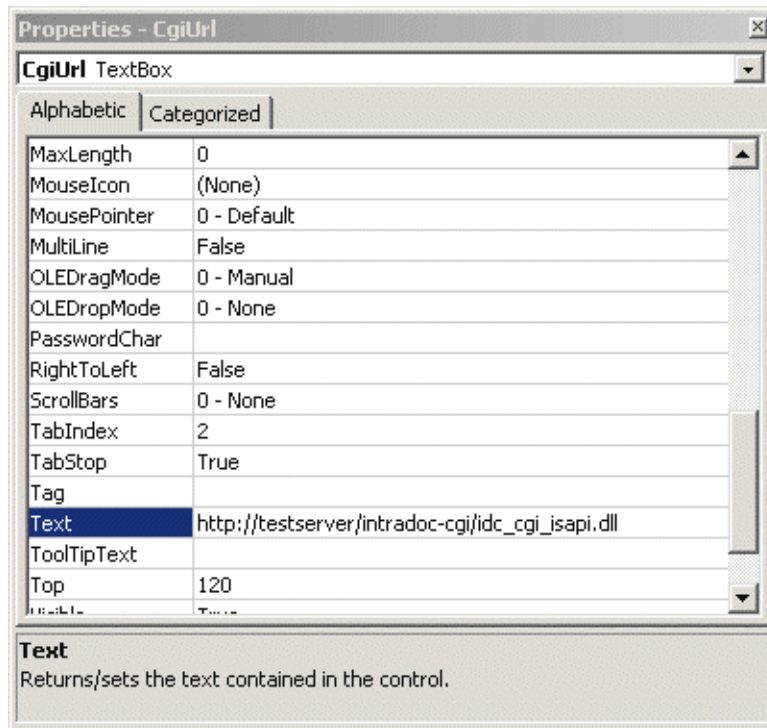
**Figure 4-2** Edited IdcClient Properties

5. On the form, draw a textbox and name it CgiUrl.
6. For the Text field, enter the HostCgiUrl value as the text to be displayed (see Figure 4-3).

For example:

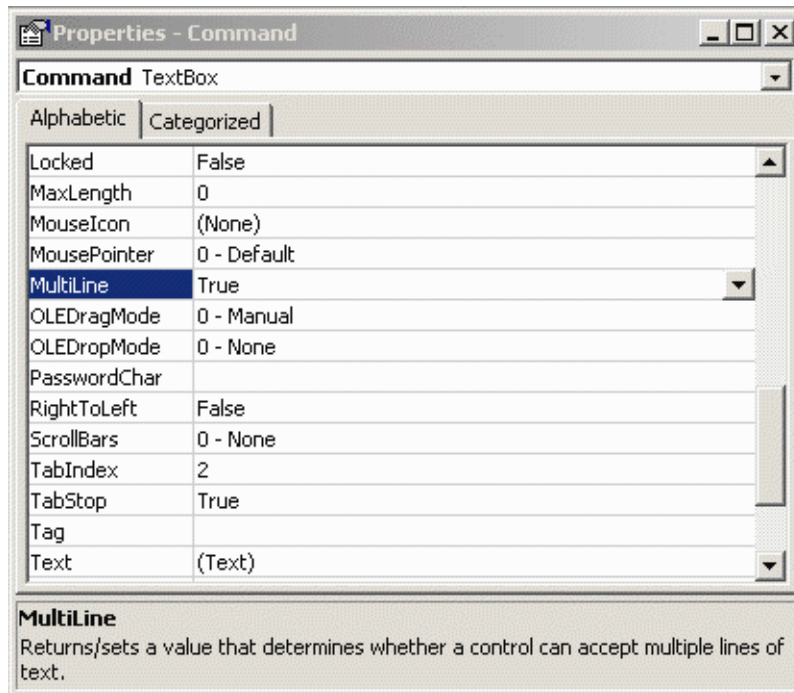
`http://testserver/intradoc-cgi/iss_idc_cgi.dll`

Figure 4-3 Edited CgiUrl TextBox propertie



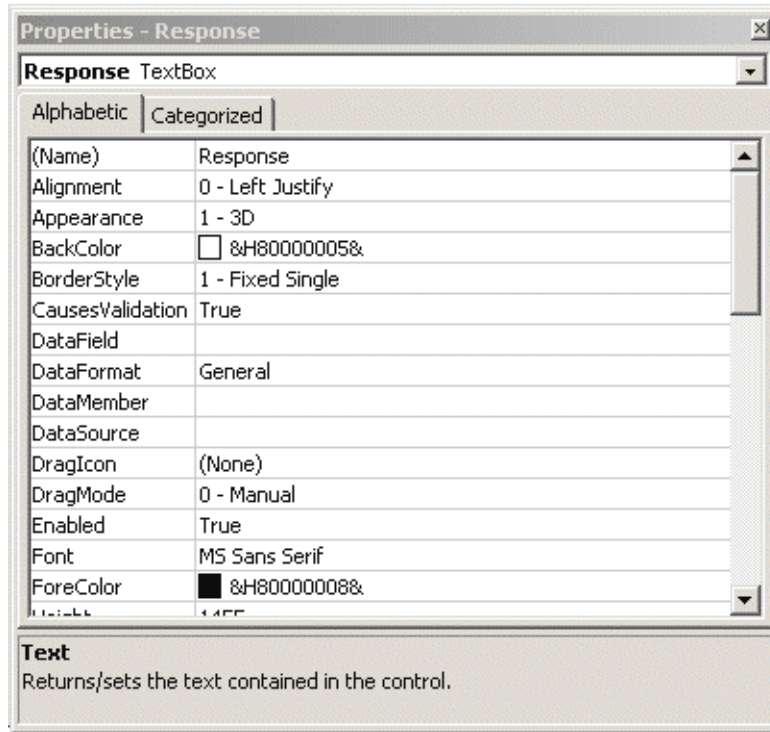
7. On the form, draw a textbox and name it Command.
8. Clear the entry for the Text field (leave blank) and set MultiLine to True (see Figure 4-4).



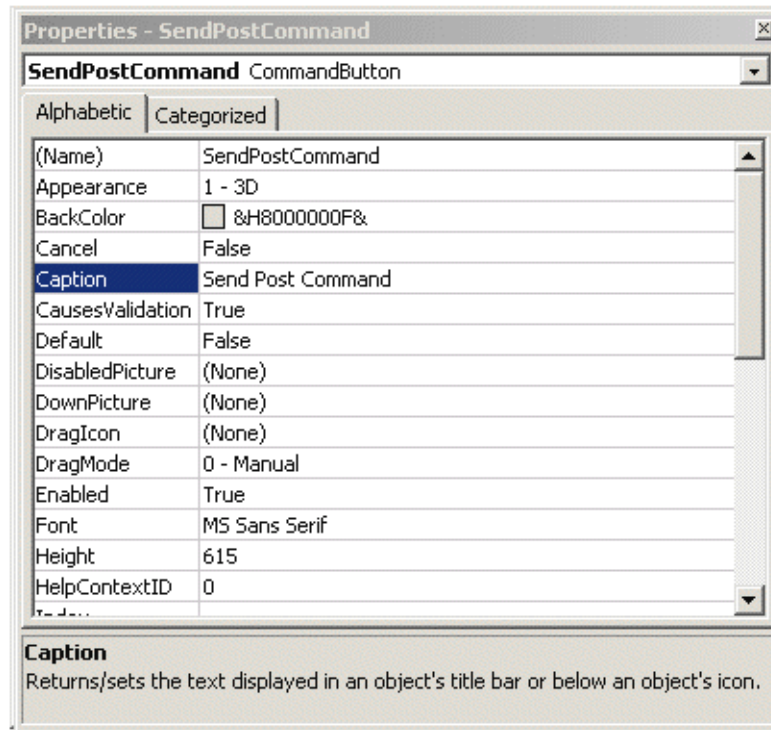
**Figure 4-4** Edited Command TextBox properties.

9. On the form, draw a textbox and name it Response.
10. Clear the entry for the Text field (leave blank).

Figure 4-5 Edited Response TextBox propertie



11. On the form, draw a button and name it SendPostCommand.
12. For the Caption field, enter "Send Post Command" as the text to be displayed (see Figure 4-6).

**Figure 4-6** Edited SendPostCommand CommandButton properties.

13. On the form, select **View—Code**.
14. Select SendPostCommand and then Click from the drop-down lists and modify the code to perform these actions (see Figure 4-7):
  - ❖ Set the Host Cgi Url
  - ❖ Issue the command
  - ❖ Optional—replace LF with CRLF to make the presentation in the edit control more readable.
  - ❖ Display the response

For example:

```
Dim R As String
IdcClientCtrl.HostCgiUrl = CgiUrl.Text
R = IdcClientCtrl.1.SendPostCommand(Command.Text)
R = Replace(R, vbLf, vbCrLf)
Response.Text = R
```

**Figure 4-7** Edited SendPostCommand\_Click code

```
Private Sub SendPostCommand_Click()  
    Dim R As String  
  
    ' Set the Host CGI Url  
    IdcClientCtrl.HostCgiUrl = CgiUrl.Text  
  
    ' Issue the command  
    R = IdcClientCtrl.SendPostCommand(Command.Text)  
  
    ' Optional--replace LF with CRLF here  
    R = Replace(R, vbLf, vbCrLf)  
  
    ' Display the response  
    Response.Text = R  
  
End Sub
```

15. Select Form and then Load from the drop-down lists and add the following lines to set the login prompt for the content server (see Figure 4-8):

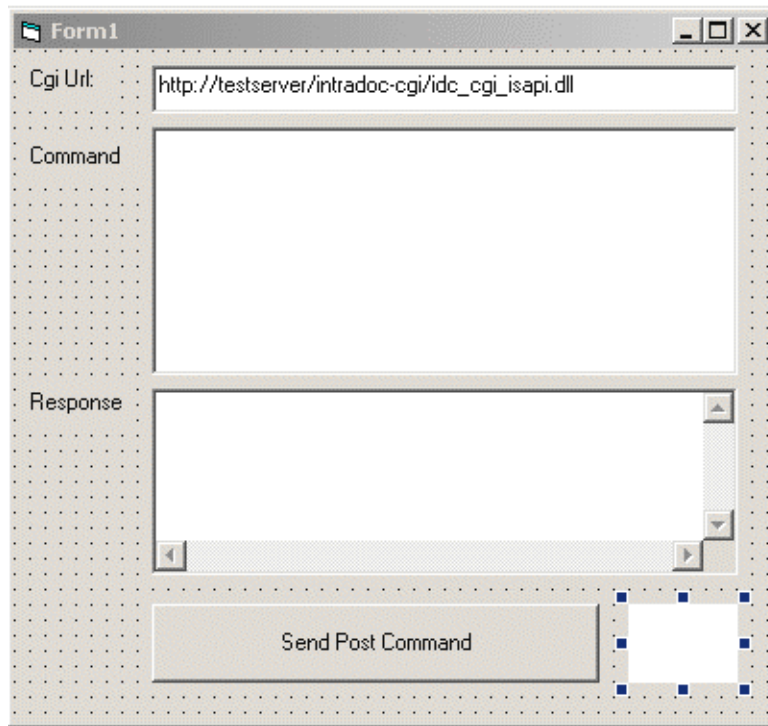
```
IdcClientCtrl.UseBrowserLoginPrompt = True  
IdcClientCtrl.UseProgressDialog = True
```

**Figure 4-8** Edited Form\_Load code

```
Private Sub Form_Load()  
    IdcClientCtrl.UseBrowserLoginPrompt = True  
    IdcClientCtrl.UseProgressDialog = True  
End Sub
```

16. Optional—add appropriate descriptive labels such as Cgi Url, Command, and Response (see Figure 4-9):

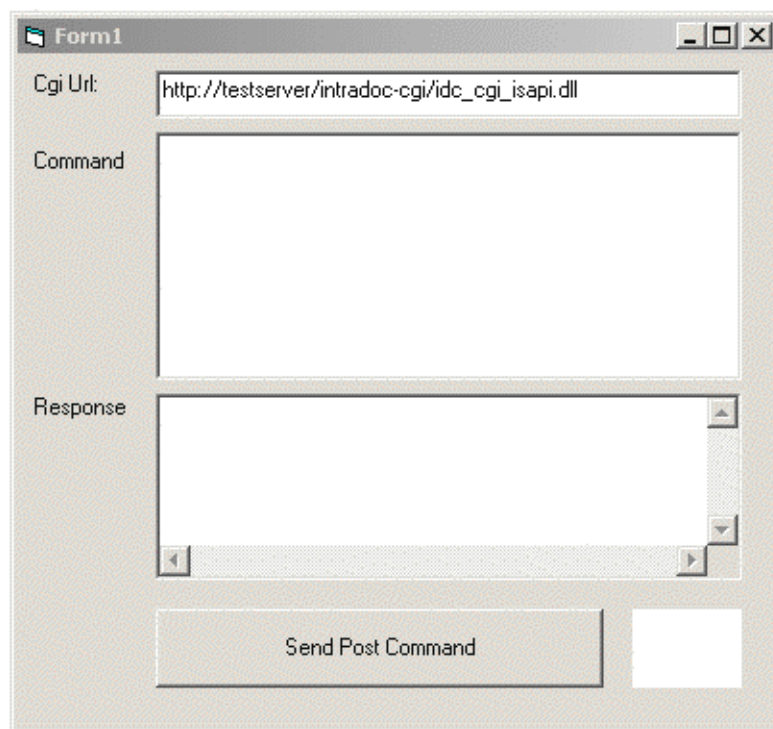
**Figure 4-9** Visual interface with descriptive label



The image shows a Windows-style window titled "Form1". Inside the window, there are three main sections. The top section is labeled "Cgi Url:" and contains a text box with the value "http://testserver/intradoc-cgi/idc\_cgi\_isapi.dll". Below this is a section labeled "Command" which is an empty text area. The third section is labeled "Response" and is a larger text area with scrollbars. At the bottom of the form, there is a button labeled "Send Post Command".

17. Select **Run—Start** to test the visual interface.

**Figure 4-10** Completed visual interfac



18. Enter a formatted command in the Command field (see Figure 4-11).

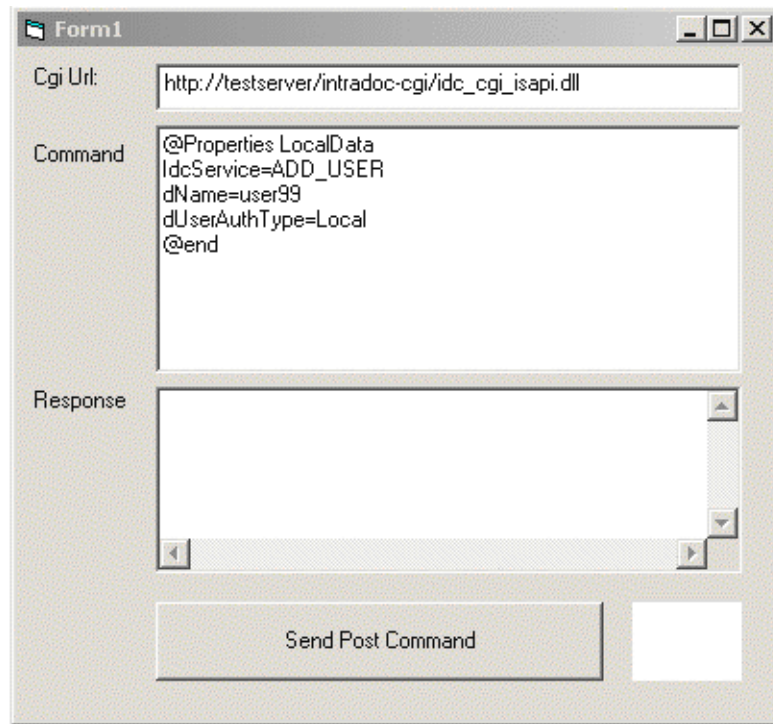
For example, this command adds a user:

```
@Properties LocalData
IdcService=ADD_USER
dName=user99
dUserAuthType=Local
@end
```



**Note:** See the *Services References Guide* for additional information on the ADD\_USER service.

**Figure 4-11** Visual interface with defined command.



The screenshot shows a window titled "Form1" with three main sections: "Cgi Url:", "Command", and "Response".

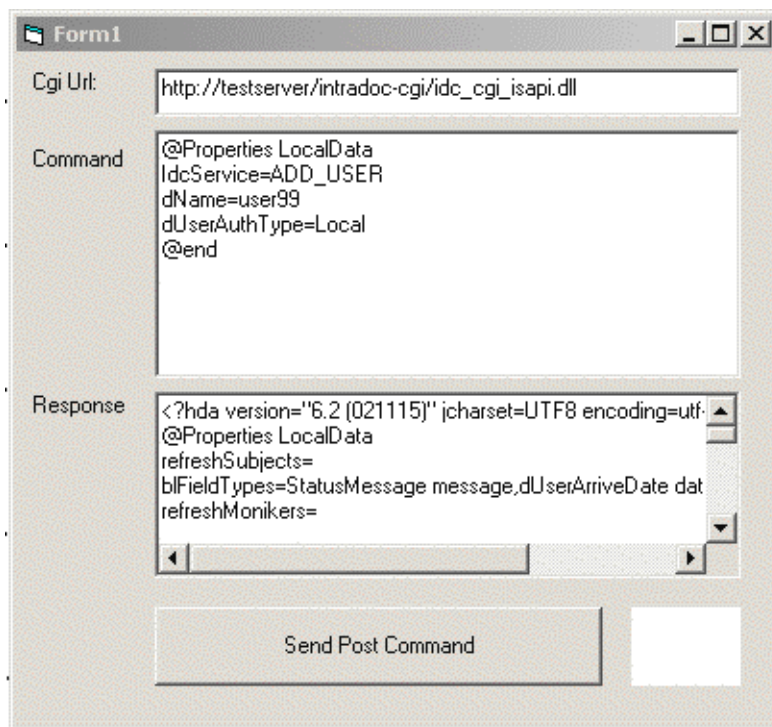
- Cgi Url:** A text box containing the URL `http://testserver/intradoc-cgi/idc.cgi_isapi.dll`.
- Command:** A text area containing the following command:

```
@Properties LocalData
IdcService=ADD_USER
dName=user99
dUserAuthType=Local
@end
```
- Response:** An empty text area with scrollbars, intended for the output of the command.

At the bottom of the window, there is a button labeled "Send Post Command" and a small empty square box to its right.

19. Click the **Send Post Command** button to execute the command. The returned results are displayed in the Response field.(see Figure 4-12).

Figure 4-12 Visual interface with returned results



### Verify the Command

20. In a web browser, login to the content server as the administrator.
21. Select **Administration—Admin Applets**.
22. Click the **User Admin** link. The applet launches and displays the added user (e.g., user99).

## IDCCLIENT EVENTS

Events are executed when the user or server performs an action. The following are IdcClient OCX Events:

- ❖ [IntradocBeforeDownload](#) (page 4-17)
- ❖ [IntradocBrowserPost](#) (page 4-17)
- ❖ [IntradocBrowserStateChange](#) (page 4-17)



- ❖ [IntradocRequestProgress](#) (page 4-18)
- ❖ [IntradocServerResponse](#) (page 4-18)

## IntradocBeforeDownload

---

Executes before a file is downloaded.

- ❖ Initiates the server actions and updates required prior to a download.

### **Parameters**

The event passes these parameters:

- ❖ ByVal params As String
- ❖ cancelDownload As Boolean

## IntradocBrowserPost

---

Executes every time a form is submitted from within a browser.

### **Parameters**

The event passes these parameters:

- ❖ ByVal url As String
- ❖ ByVal params As String
- ❖ cancelPost As Boolean

## IntradocBrowserStateChange

---

Executes whenever the browser state changes.

### **Parameters**

The event passes these parameters:

- ❖ ByVal browserStateItem As String
- ❖ ByVal enabled As Boolean

## IntradocRequestProgress

---

Executes a request for a progress report to be sent from the server. This event only occurs after a method has been called.

### **Parameters**

The event passes these parameters:

- ❖ ByVal `statusData` As String
- ❖ ByVal `isDone` As Boolean

## IntradocServerResponse

---

Executes after the server completes a requested action. For example, after a file has been downloaded. This event handles HDA encoded data that is a response from the server. This event only occurs when an action is performed in the browser.

### **Parameters**

The event passes one parameter:

- ❖ ByVal `response` As String

## IDCCLIENT METHODS

---

The following IdcClient OCX Methods are available:

- |  |  |
|--|--|
| ❖ <a href="#">AboutBox</a> (page 4-19)             | ❖ <a href="#">NavigateCgiPage</a> (page 4-27)      |
| ❖ <a href="#">Back</a> (page 4-19)                 | ❖ <a href="#">RefreshBrowser</a> (page 4-28)       |
| ❖ <a href="#">CancelRequest</a> (page 4-20)*       | ❖ <a href="#">SendCommand</a> (page 4-28)*         |
| ❖ <a href="#">DoCheckoutLatestRev</a> (page 4-20)  | ❖ <a href="#">SendPostCommand</a> (page 4-28)*     |
| ❖ <a href="#">DownloadFile</a> (page 4-21)         | ❖ <a href="#">SetFocus</a> (page 4-29)             |
| ❖ <a href="#">DownloadNativeFile</a> (page 4-21)   | ❖ <a href="#">ShowDMS</a> (page 4-30)              |
| ❖ <a href="#">Drag</a> (page 4-22)                 | ❖ <a href="#">ShowDocInfoLatestRev</a> (page 4-30) |
| ❖ <a href="#">EditDocInfoLatestRev</a> (page 4-23) | ❖ <a href="#">ShowWhatsThis</a> (page 4-30)        |

- ❖ [Forward](#) (page 4-23)
- ❖ [GoCheckinPage](#) (page 4-24)
- ❖ [Home](#) (page 4-25)
- ❖ [InitiateFileDownload](#) (page 4-25)\*
- ❖ [InitiatePostCommand](#) (page 4-26)\*
- ❖ [Move](#) (page 4-26)
- ❖ [Navigate](#) (page 4-27)
- ❖ [StartSearch](#) (page 4-31)
- ❖ [Stop](#) (page 4-31)
- ❖ [UndoCheckout](#) (page 4-32)
- ❖ [ViewDocInfo](#) (page 4-32)
- ❖ [ViewDocInfoLatestRev](#) (page 4-33)
- ❖ [ZOrder](#) (page 4-33)

Methods marked with an asterisk (\*) are ones which are not related to browser activity and which return a value.



**Important:** All parameters are required unless otherwise indicated.

## AboutBox

---

Sub AboutBox()

### **Description**

Launches the About box containing product version information.

- ❖ This method displays the product About box.
- ❖ The method returns FALSE if the call cannot be executed.

### **Parameters**

None

## Back

---

Sub Back()

### **Description**

Displays the previous HTML page.

- ❖ Returns the user to the previous screen.

- ❖ The method retrieves the previous HTML page from cached information for display to the user.

### **Parameters**

None

## **CancelRequest**

---

Function CancelRequest() As Boolean

### **Description**

This method cancels the currently active request. Returns FALSE if the function is unable to cancel the request or if there is no request currently active.

### **Parameters**

None

### **Output**

Returns a Boolean value:

- ❖ Returns TRUE if request is cancelled.
- ❖ Returns FALSE if the cancel request is not performed.

## **DoCheckoutLatestRev**

---

Sub DoCheckoutLatestRev(docName As String, curID As String)

### **Description**

Checks out or locks the latest content item revision.

- ❖ Given a content item name and the version label, the method checks out the latest content item revision.
- ❖ Executes the IntradocServerResponse event. The event is executed before the method occurs. See [IdcClient Events](#) (page 4-16) for details.



**Note:** The `curID` is the content item version label, not the generated content item revision ID.

This function returns the following:

- ❖ Serialized HDA containing dID and dDocName.
- ❖ FALSE if the latest revision cannot be checked out or cannot be found in the system.
- ❖ The data that was passed in as parameters.

### **Parameters**

- ❖ docName: The user-assigned content item name.
- ❖ curID: The unique identifier for the latest revision. *Optional*.

## **DownloadFile**

---

Function DownloadFile(command As String, filename As String) As String

### **Description**

Downloads the defined file.

- ❖ Given a currently-associated command and the file type, this method performs a file download of the post-conversion file (compare DownloadNativeFile).
- ❖ Executes the IntradocBeforeDownload event. The event is executed before the method occurs. See [IdcClient Events](#) (page 4-16) for details.

This function returns the following:

- ❖ Serialized HDA containing the status code and status method.
- ❖ The data that was passed in as parameters.
- ❖ FALSE if it is unable to download the specified file.

### **Parameters**

- ❖ command: The currently-associated command.
- ❖ filename: The file format. This is the file type such as PDF, HTM, or other supported format.

## **DownloadNativeFile**

---

Function DownloadNativeFile(id As String, docName As String, filename As String) As String

### **Description**

Downloads the defined native file.

- ❖ Given a content item revision ID, a content item name, and a file type, this method performs a file download of the native file (compare `DownloadFile`).
- ❖ Executes the `IntradocBeforeDownload` event. The event is executed before the method occurs. See [IdcClient Events](#) (page 4-16) for details.



**Note:** The `id` is the generated content item revision ID, not the content item version label.

This function returns the following:

- ❖ Serialized HDA containing `dID` and `dDocName`.
- ❖ The data that was passed in as parameters.
- ❖ `FALSE` if it is unable to download the specified file.

### **Parameters**

- ❖ `id`: The unique identifier for the latest revision.
- ❖ `docName`: The user-assigned content item name.
- ❖ `filename`: The file format. This is the file type such as `DOC`, `RTF`, or any other supported format.

## **Drag**

---

```
Sub Drag([nAction])
```

### **Description**

Begins, ends, or cancels a drag operation.

- ❖ The `Drag` method is handled the same as a `Standard Control` implementation.
- ❖ Refer to a Visual Basic API reference for additional information.

### **Parameters**

- ❖ `nAction`: Indicates the action to perform. If you omit `nAction`, `nAction` is set to `1`.

The settings for the `Drag` method are:

- ❖ 0: Cancel drag operation; restore original position of control.
- ❖ 1: (Default) Begin dragging the control.
- ❖ 2: End dragging — drop the control.

## EditDocInfoLatestRev

---

```
Sub EditDocInfoLatestRev(docName As String, curID As String, activateAction As String)
```

### Description

Edits the content item information for the latest revision.

- ❖ ODMA related.
- ❖ Given a content item name, the version label, and the currently-active requested action, the method edits the content item information for the latest revision.
- ❖ The function returns FALSE if the content item information for the latest revision cannot be edited or cannot be found in the system.



**Note:** The `curID` is the content item version label, not the generated content item revision ID.

### Parameters

- ❖ `curID`: The unique identifier for the latest revision.
- ❖ `activateAction`: Passed to `ODMActivate`. This can be used as `Idoc Script`. *Optional*.
- ❖ `docName`: The user-assigned content item name. *Optional*.

## Forward

---

```
Sub Forward()
```

### Description

Displays the next HTML page.

- ❖ Moves the user to the next screen.
- ❖ This method retrieves cached information for the next HTML page for display to the user.

## **Parameters**

None

## **GoCheckinPage**

---

```
Sub GoCheckinPage(id As String, docName As String, isNew As Boolean, params As String)
```

## **Description**

Checks in a new content item or a content item revision.

- ❖ Given the content item revision ID and the content item name, the function checks in a new content item or a content item revision.
- ❖ This method opens the content item check-in page and enters the unique content item identifier, user-assigned content item name, and any assigned content item parameters into the associated text fields. It is also specified whether this is a new content item or a revision.



**Note:** The `id` is the generated content item revision ID, not the content item version label.

This function returns the following:

- ❖ FALSE if it is unable to check in the specified file.
- ❖ Serialized HDA containing `dID` and `dDocName`.
- ❖ The data that was passed in as parameters.

## **Parameters (all optional)**

- ❖ `id`: The unique identifier for the latest revision.
- ❖ `docName`: The user-assigned content item name.
- ❖ `IsNew`: Defines whether the content item to be checked in is a new content item or a revision.
  - If TRUE, a new unique content item version label is assigned.
  - Default is TRUE.
- ❖ `params`: The parameters that pre-fill the Check In page.



## Home

---

Sub Home()

### **Description**

Returns the user to the defined home page.

- ❖ Moves the user to the home screen.
- ❖ Executes an HTML page request and displays the defined home page to the user.

### **Parameters**

None

## InitiateFileDownload

---

Function InitiateFileDownload(command As String, filename As String) As String

### **Description**

Initiates a file download.

- ❖ Given the currently-associated command and the file type, the function initiates a file download. This method initiates a file download of a specific rendition of a content item, the latest revision, or the latest released revision.
- ❖ Executes the `IntradocServerResponse` event. The event is executed before the method occurs.
- ❖ See [IdcClient Events](#) (page 4-16) for details.

### **Parameters**

- ❖ `command`: The currently-associated command.
- ❖ `filename`: The file format. This is the file type such as PDF, HTM, or other supported format.

### **Output**

- ❖ Returns serialized HDA containing the requested information.
- ❖ Returns the data that was passed in as parameters.

## InitiatePostCommand

---

Function InitiatePostCommand(postData As String) As String

### **Description**

Initiates a post command.

- ❖ Initiates a service call. Given assigned post data, this method initiates a post command.
- ❖ Executes the `IntradocServerResponse` event. The event is executed before the method occurs. See [IdcClient Events](#) (page 4-16) for details.

### **Parameters**

- ❖ `postData`: The serialized HDA containing the service command and any necessary service parameters.

### **Output**

- ❖ Returns serialized HDA containing the requested information.
- ❖ Returns `StatusCode` and `StatusMessage`.
  - The `StatusCode` will be negative if a failure occurs, and `StatusMessage` will indicate the error.
  - If the returned HDA does not contain a `StatusCode` parameter, the service call succeeded.

## Move

---

Sub Move(Left As Single, [Top], [Width], [Height])

### **Description**

Moves an object.

- ❖ The `Move` method is handled the same as a Standard Control implementation.
- ❖ Refer to a Visual Basic API reference for additional information.

### **Parameters**

- ❖ nLeft: Specifies the horizontal coordinate for the left edge of the object. This is a single-precision value.
- ❖ nTop: Specifies the vertical coordinate for the top edge of the object. This is a single-precision value.
- ❖ nWidth: Specifies the new width of the object. This is a single-precision value.
- ❖ nHeight: Specifies the new height of the object. This is a single-precision value.

## **Navigate**

---

Sub Navigate(url As String)

### **Description**

Computes the URL path.

- ❖ Given a complete URL, this method computes the URL from the serialized HDA and returns the value as a string.

This function returns the following:

- ❖ Serialized HDA containing the requested information.
- ❖ The data that was passed in as parameters.

### **Parameters**

- ❖ url: The complete URL path.

## **NavigateCgiPage**

---

Sub NavigateCgiPage(params As String)

### **Description**

Computes the CGI path.

- ❖ Given defined content item parameters, this method computes the CGI path from the serialized HDA and returns the value as a string.

### ***Parameters***

- ❖ params: The assigned content item parameters.

## **RefreshBrowser**

---

Sub RefreshBrowser()

### ***Description***

Refreshes the browser.

- ❖ This method refreshes the web browser and updates dynamic information.

### ***Parameters***

None

## **SendCommand**

---

Function SendCommand(params As String) As String

### ***Description***

Issues a service request to the content server.

- ❖ Given defined content item parameters, the function executes a service from the content server related to content item handling.

### ***Parameters***

- ❖ params: The CGI URL encoded parameters.

### ***Output***

- ❖ Returns serialized HDA containing the requested information.
- ❖ Returns the data that was passed in as parameters.

## **SendPostCommand**

---

Function SendPostCommand(postData As String) As String

## **Description**

Sends a post command.

- ❖ Executes a service call.
- ❖ Executes the `IntradocBrowserPost` event. The event is executed before the method occurs. See [IdcClient Events](#) (page 4-16) for details.

## **Parameters**

- ❖ `postData`: The serialized HDA containing the service command and any necessary service parameters.

## **Output**

- ❖ Returns serialized HDA containing the requested information.
- ❖ Returns `StatusCode` and `StatusMessage`.
  - The `StatusCode` will be negative if a failure occurs, and `StatusMessage` will indicate the error.
  - If the returned HDA does not contain a `StatusCode` parameter, the service call succeeded.

## **SetFocus**

---

Sub `SetFocus()`

### **Description**

Assigns the focus to a control.

- ❖ The `SetFocus` method is handled the same as a Standard Control implementation.
- ❖ Refer to a Visual Basic API reference for additional information.

### **Parameters**

None

## ShowDMS

---

Sub ShowDMS()

### **Description**

Opens the HTML page associated with the Content Manager.

- ❖ ODMA related.
- ❖ Displays the Content Manager access page in a browser.

### **Parameters**

None

## ShowDocInfoLatestRev

---

Sub ShowDocInfoLatestRev(docName As String, curID As String, activateAction As String)

### **Description**

Displays the content item information for the latest revision.

- ❖ Given a content item name, the version label, and the action to perform, the function displays the content item information for the latest revision in the browser control.



**Note:** The `curID` is the content item version label, not the generated content item revision ID.

### **Parameters**

- ❖ `docName`: The user-assigned content item name.
- ❖ `curID`: The unique identifier for the latest revision. *Optional*.
- ❖ `activateAction`: The currently-active requested action. *Optional*

## ShowWhatsThis

---

Sub ShowWhatsThis()

**Description**

Displays the What's This Help topic specified for an object with the WhatsThisHelpID property.

- ❖ The ShowWhatsThis method is handled the same as a Standard Control implementation.
- ❖ Refer to a Visual Basic API reference for additional information.

**Parameters**

- ❖ Object: Specifies the object for which the What's This Help topic is displayed.

**StartSearch**

---

```
Sub StartSearch()
```

**Description**

Displays the query page in the browser control.

- ❖ Performs browser manipulation.

**Parameters**

None

**Stop**

---

```
Sub Stop()
```

**Description**

Stops the browser.

- ❖ This method stops or cancels the loading of information in the browser.

**Parameters**

None

## UndoCheckout

---

```
Sub UndoCheckout(docName As String, curID As String)
```

### **Description**

This service reverses a content item checkout.

- ❖ Given a content item name and a version label, this service attempts to locate the content item in the system and undo the check out. The service fails if the content item does not exist in the system, if the content item is not checked out or the user does not have sufficient privilege to undo the checkout.
- ❖ Executes the `IntradocServerResponse` event. The event is executed before the method occurs.
- ❖ See [IdcClient Events](#) (page 4-16) for details.



**Note:** The `curID` is the content item version label, not the generated content item revision ID.

### **Parameters**

- ❖ `curID`: The unique identifier for the latest revision.
- ❖ `docName`: The user-assigned content item name. *Optional.*

## ViewDocInfo

---

```
Sub ViewDocInfo(id As String)
```

### **Description**

Navigates to the content item information page and displays content item information in a browser.

- ❖ Performs browser manipulation.
- ❖ Given a content item revision ID, the method displays content item information in a browser.



**Note:** The `id` is the generated content item revision ID, not the content item version label.



### **Parameters**

- ❖ `id`: The unique identifier for the latest revision.

## **ViewDocInfoLatestRev**

---

```
Sub ViewDocInfoLatestRev(docName As String, curID As String)
```

### **Description**

Navigates to the content item information page and displays content item information for the latest revision.

- ❖ Given a content item name and a version label, the method displays the content item information for the latest revision.



**Note:** The `curID` is the content item version label, not the generated content item revision ID.

This function returns the following:

- ❖ Serialized HDA containing `dID` and `dDocName`.
- ❖ The data that was passed in as parameters.

### **Parameters**

- ❖ `docName`: The user assigned content item name.
- ❖ `curID`: The unique identifier for the latest revision.

## **ZOrder**

---

```
Sub ZOrder([Position])
```

### **Description**

Places a specified form or control at the front or back of the z-order within its graphical level.

- ❖ The `ZOrder` method is handled the same as a Standard Control implementation.
- ❖ Refer to a Visual Basic API reference for additional information.

### **Parameters**

- ❖ `nOrder`: Specifies an integer indicating the position of the object relative to other objects. If you omit `nOrder`, the setting is 0.

The settings for the `ZOrder` method are:

- ❖ 0: (Default) The object is positioned at the front of the z-order.
- ❖ 1: The object is positioned at the back of the z-order.

## **IDCCCLIENT PROPERTIES**

---

Each data item or “attribute” is implemented as a “Property” in Visual Basic. Properties are exposed through the Public Interface of an object within the Visual Basic development environment. These attributes can be used to further describe elements.

These are the IdcClient OCX Properties:

- ❖ [ClientControlledContextValue](#) (page 4-34)
- ❖ [HostCgiUrl](#) (page 4-34)
- ❖ [Password](#) (page 4-35)
- ❖ [UseBrowserLoginPrompt](#) (page 4-35)
- ❖ [UseProgressDialog](#) (page 4-35)
- ❖ [UserName](#) (page 4-35)
- ❖ [WorkingDir](#) (page 4-36)

### **ClientControlledContextValue**

---

Provides the user-supplied context value. This value becomes available to Idoc Script as the variable `ClientControlled` in any web page delivered by the content server.

- ❖ Returns the value as a string.
- ❖ Takes no parameters.

### **HostCgiUrl**

---

Provides the complete URL path of the host CGI bin.

- ❖ Returns the value as a string.
- ❖ Takes no parameters.

## Password

---

Provides the assigned user password.

- ❖ Returns the value as a string.
- ❖ Takes no parameters.

## UseBrowserLoginPrompt

---

Allows the use of a browser login prompt. Defines whether a dialog box for user authentication will display.

- ❖ If set to TRUE, control will open a dialog box for user authentication
- ❖ Default is TRUE.

Returns a Boolean value:

- ❖ Returns TRUE if the login was successful.
- ❖ Returns FALSE if the login was denied.

## UseProgressDialog

---

Allows the use of a user progress dialog. Defines whether a dialog box for user authentication will display.

- ❖ If set to TRUE, control will open a dialog box for user progress.
- ❖ Default is TRUE.

Returns a Boolean value:

- ❖ Returns TRUE if the action was completed.
- ❖ Returns FALSE if the action failed.

## UserName

---

Provides the assigned user name.

- ❖ Returns the value as a string.
- ❖ Takes no parameters.

## **WorkingDir**

---

Specifies the working directory as a full path. This is the location where downloaded files are placed.

- ❖ Returns the value as a string.
- ❖ Takes no parameters.



# THIRD PARTY LICENSES

## OVERVIEW

---

This appendix includes a description of the Third Party Licenses for all the third party products included with this product.

- ❖ [Apache Software License](#) (page A-1)
- ❖ [W3C® Software Notice and License](#) (page A-2)
- ❖ [Zlib License](#) (page A-3)
- ❖ [General BSD License](#) (page A-4)
- ❖ [General MIT License](#) (page A-5)
- ❖ [Unicode License](#) (page A-5)
- ❖ [Miscellaneous Attributions](#) (page A-7)

## APACHE SOFTWARE LICENSE

---

- \* Copyright 1999-2004 The Apache Software Foundation.
- \* Licensed under the Apache License, Version 2.0 (the "License");
- \* you may not use this file except in compliance with the License.
- \* You may obtain a copy of the License at
- \* <http://www.apache.org/licenses/LICENSE-2.0>
- \*

- \* Unless required by applicable law or agreed to in writing, software
- \* distributed under the License is distributed on an "AS IS" BASIS,
- \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
- \* See the License for the specific language governing permissions and
- \* limitations under the License.

## **W3C® SOFTWARE NOTICE AND LICENSE**

---

- \* Copyright © 1994-2000 World Wide Web Consortium,
- \* (Massachusetts Institute of Technology, Institut National de
- \* Recherche en Informatique et en Automatique, Keio University).
- \* All Rights Reserved. <http://www.w3.org/Consortium/Legal/>
- \*
- \* This W3C work (including software, documents, or other related items) is
- \* being provided by the copyright holders under the following license. By
- \* obtaining, using and/or copying this work, you (the licensee) agree that
- \* you have read, understood, and will comply with the following terms and
- \* conditions:
- \*
- \* Permission to use, copy, modify, and distribute this software and its
- \* documentation, with or without modification, for any purpose and without
- \* fee or royalty is hereby granted, provided that you include the following
- \* on ALL copies of the software and documentation or portions thereof,
- \* including modifications, that you make:
- \*
- \* 1. The full text of this NOTICE in a location viewable to users of the
- \* redistributed or derivative work.
- \*
- \* 2. Any pre-existing intellectual property disclaimers, notices, or terms
- \* and conditions. If none exist, a short notice of the following form
- \* (hypertext is preferred, text is permitted) should be used within the
- \* body of any redistributed or derivative code: "Copyright ©
- \* [\$date-of-software] World Wide Web Consortium, (Massachusetts

\* Institute of Technology, Institut National de Recherche en  
\* Informatique et en Automatique, Keio University). All Rights  
\* Reserved. <http://www.w3.org/Consortium/Legal/>  
\*  
\* 3. Notice of any changes or modifications to the W3C files, including the  
\* date changes were made. (We recommend you provide URIs to the location  
\* from which the code is derived.)  
\*  
\* THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS  
\* MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT  
\* NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR  
\* PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE  
\* ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.  
\*  
\* COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR  
\* CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR  
\* DOCUMENTATION.  
\*  
\* The name and trademarks of copyright holders may NOT be used in advertising  
\* or publicity pertaining to the software without specific, written prior  
\* permission. Title to copyright in this software and any associated  
\* documentation will at all times remain with copyright holders.  
\*

## ZLIB LICENSE

---

\* zlib.h -- interface of the 'zlib' general purpose compression library  
version 1.2.3, July 18th, 2005

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied  
warranty. In no event will the authors be held liable for any damages

arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly [jloup@gzip.org](mailto:jloup@gzip.org)

Mark Adler [madler@alumni.caltech.edu](mailto:madler@alumni.caltech.edu)

## GENERAL BSD LICENSE

---

Copyright (c) 1998, Regents of the University of California

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

"Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

"Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

"Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.



THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## GENERAL MIT LICENSE

---

Copyright (c) 1998, Regents of the Massachusetts Institute of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## UNICODE LICENSE

---

UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

Unicode Data Files include all data files under the directories <http://www.unicode.org/Public/>, <http://www.unicode.org/reports/>, and <http://www.unicode.org/cldr/data/>. Unicode Software includes any source code published in the Unicode Standard or under the directories <http://www.unicode.org/Public/>, <http://www.unicode.org/reports/>, and <http://www.unicode.org/cldr/data/>.

## Third Party Licenses

NOTICE TO USER: Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

### COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2006 Unicode, Inc. All rights reserved. Distributed under the Terms of Use in <http://www.unicode.org/copyright.html>.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.

THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

Unicode and the Unicode logo are trademarks of Unicode, Inc., and may be registered in some jurisdictions. All other trademarks and registered trademarks mentioned herein are the property of their respective owners

## MISCELLANEOUS ATTRIBUTIONS

---

Adobe, Acrobat, and the Acrobat Logo are registered trademarks of Adobe Systems Incorporated.

FAST Instream is a trademark of Fast Search and Transfer ASA.

HP-UX is a registered trademark of Hewlett-Packard Company.

IBM, Informix, and DB2 are registered trademarks of IBM Corporation.

Jaws PDF Library is a registered trademark of Global Graphics Software Ltd.

Kofax is a registered trademark, and Ascent and Ascent Capture are trademarks of Kofax Image Products.

Linux is a registered trademark of Linus Torvalds.

Mac is a registered trademark, and Safari is a trademark of Apple Computer, Inc.

Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation.

MrSID is property of LizardTech, Inc. It is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Oracle is a registered trademark of Oracle Corporation.

Portions Copyright © 1994-1997 LEAD Technologies, Inc. All rights reserved.

Portions Copyright © 1990-1998 Handmade Software, Inc. All rights reserved.

Portions Copyright © 1988, 1997 Aladdin Enterprises. All rights reserved.

Portions Copyright © 1997 Soft Horizons. All rights reserved.

Portions Copyright © 1995-1999 LizardTech, Inc. All rights reserved.

Red Hat is a registered trademark of Red Hat, Inc.

Sun is a registered trademark, and Sun ONE, Solaris, iPlanet and Java are trademarks of Sun Microsystems, Inc.

Sybase is a registered trademark of Sybase, Inc.

UNIX is a registered trademark of The Open Group.

Verity is a registered trademark of Autonomy Corporation plc

## Third Party Licenses

## #

-c connection\_mode  
  auto, 2-6  
  server, 2-6  
  standalone, 2-6

## A

AboutBox, 4-19  
Active Server Page  
  Calling IdcCommandX, 3-4  
ActiveX command utility, *See methods*, 3-16  
ASP  
  Calling IdcCommandX, 3-4  
Attributes  
  ClientControlledContextValue, 4-34  
  HostCgiUrl, 4-34  
  Password, 4-35  
  UseBrowserLoginPrompt, 4-35  
  UseProgressDialog, 4-35  
  UserName, 4-35  
  WorkingDir, 4-36

## B

Back, 4-19

## C

Calling IdcCommandX from an Active Server Page, 3-4  
calling services remotely, 2-7  
CancelRequest, 4-20  
ClientControlledContextValue, 4-34  
closeServerConnection(), 3-17  
command file syntax, 2-2  
  precedence, 2-4  
  special characters  
    , 2-4  
    #, 2-4  
    EOD, 2-4  
  special tags  
    IdcService=, 2-4  
command line options, 2-2  
computeNativeFilePath(Data As String) as string, 3-18  
computeURL(Data As String, IsAbsolute As Boolean) as string, 3-19  
computeWebFilePath(Data As String) as string, 3-22  
ConnectToServer() as boolean  
  returns, 3-23  
connectToServer() as boolean, 3-23

creating and executing IdcCommand parameters, 2-7,  
  2-7

## D

DoCheckoutLatestRev, 4-20  
DownloadFile, 4-21  
DownloadNativeFile, 4-21  
Drag, 4-22

## E

EditDocInfoLatestRev, 4-23  
EOD, 2-4  
Events  
  IntradocBeforeDownload, 4-17  
  IntradocBrowserPost, 4-17  
  IntradocBrowserStateChange, 4-17  
  IntradocRequestProgress, 4-18  
  IntradocServerResponse, 4-18  
executeCommand (Data As String), 3-24  
  parameters  
    Data, 3-24  
  returns, 3-24

## F

Forward, 4-23

## G

GoCheckinPage, 4-24

## H

HostCgiUrl, 4-34

## I

IdcClient ActiveX Control, 4-5  
IdcClient Events, 4-16  
IdcClient Methods, 4-18  
IdcClient OCX Component, 4-1  
IdcClient Properties, 4-34  
IdcCommand  
  repository server Command Utility, 2-7  
  calling services remotely, 2-7  
  command file syntax, 2-2  
  command line options, 2-2  
  using the launcher, 2-17

IdcCommandX and ASP, 3-4  
IdcCommandX-repository server ActiveX Command  
    Utility  
    methods, 3-16  
IdcService= (*command file syntax tag*), 2-4  
init as boolean (*IntradocDir As String*)  
    parameters  
        UserName, 3-27  
init as boolean (*StellentDir As String*)  
    parameters  
        StellentDir, 3-27  
InitiateFileDownload, 4-25  
InitiatePostCommand, 4-26  
IntradocBeforeDownload, 4-17  
IntradocBrowserPost, 4-17  
IntradocBrowserStateChange, 4-17  
IntradocDir As String (*init as boolean*), 3-27  
IntradocRequestProgress, 4-18  
IntradocServerResponse, 4-18

## M

### Methods

AboutBox, 4-19  
Back, 4-19  
CancelRequest, 4-20  
DoCheckoutLatestRev, 4-20  
DownloadFile, 4-21  
DownloadNativeFile, 4-21  
Drag, 4-22  
EditDocInfoLatestRev, 4-23  
Forward, 4-23  
GoCheckinPage, 4-24  
InitiateFileDownload, 4-25  
InitiatePostCommand, 4-26  
Move, 4-26  
Navigate, 4-27  
NavigateCgiPage, 4-27  
RefreshBrowser, 4-28  
SendCommand, 4-28  
SendPostCommand, 4-28  
SetFocus, 4-29  
ShowDocInfoLatestRev, 4-30  
ShowWhatsThis, 4-30  
StartSearch, 4-31  
Stop, 4-31  
UndoCheckout, 4-32  
ViewDocInfo, 4-32  
ViewDocInfoLatestRev, 4-33  
Zorder, 4-33  
methods (*ActiveX command utility*), 3-16  
    closeServerConnection(), 3-17

    computeNativeFilePath(Data As String) as  
        string, 3-18  
    computeURL(Data As String, IsAbsolute As Boolean)  
        as string, 3-19  
    computeWebFilePath(Data As String) as string, 3-22  
    connectToServer() as boolean, 3-23  
    executeCommand (Data As String), 3-24  
    init as boolean (IntradocDir As String), 3-27  
Microsoft Visual Basic, 4-5, 4-5  
Move, 4-26

## N

Navigate, 4-27  
NavigateCgiPage, 4-27

## O

OCX Component, 4-1  
Overview  
    Audience, 1-2  
    Conventions, 1-2

## P

Password, 4-35  
precedence, 2-4

## R

RefreshBrowser, 4-28

## S

SendCommand, 4-28  
SendPostCommand, 4-28  
SetFocus, 4-29  
ShowDocInfoLatestRev, 4-30  
ShowWhatsThis, 4-30  
StartSearch, 4-31  
Stop, 4-31

## U

UndoCheckout, 4-32  
UseBrowserLoginPrompt, 4-35  
UseProgressDialog, 4-35  
UserName, 4-35  
using the Launcher, 2-17

## **V**

ViewDocInfo, 4-32  
ViewDocInfoLatestRev, 4-33  
Visual Basic, 4-5, 4-5

## **W**

WorkingDir, 4-36

## **Z**

ZOrder, 4-33

