

iMacros Version 5

A unique solution for
Web Automation, Data Extraction & Web Testing

Patents Pending

Table of Contents

| | | |
|-----------------|--|-----------|
| Part I | Introducing iMacros | 7 |
| 1 | Why iMacros? | 8 |
| Part II | Using iMacros | 9 |
| 1 | Start iMacros | 9 |
| 2 | The iMacro Graphical User Interface | 9 |
| 3 | Recording | 11 |
| 4 | Replay | 12 |
| 5 | Edit Macros | 12 |
| 6 | Security | 13 |
| 7 | Saving Web Sites | 14 |
| 8 | Web Site Screenshot | 14 |
| 9 | Save Web Site Elements | 16 |
| | Download Files | 16 |
| | Save Item | 16 |
| | Save Target As | 17 |
| | Save Picture As | 17 |
| 10 | Dialog Manager | 18 |
| | Login | 18 |
| | Javascript | 19 |
| | Web Page Dialogs | 19 |
| | Print | 20 |
| | Security | 20 |
| | Certificates | 20 |
| | Page Errors | 21 |
| 11 | Print | 21 |
| 12 | Offline | 22 |
| Part III | Advanced | 22 |
| 1 | Tabbed Browser | 22 |
| 2 | Frames | 23 |
| 3 | Fine Tune TAG Commands | 24 |
| 4 | Variables | 25 |
| 5 | Proxy Server | 26 |
| 6 | Submitting Multiple Datasets to Web Sites | 27 |
| | Input from Comma Separated Data (CSV) File | 27 |
| | Input from List of Variables File | 28 |
| | Input from Database | 29 |
| 7 | Extract Data from Websites | 30 |

| | |
|---|-----------|
| Extract single elements | 30 |
| The POS Parameter..... | 33 |
| Extract with relative Positioning..... | 34 |
| Using wildcards..... | 35 |
| Example: Keyword Anchor..... | 35 |
| Example: Data separated with | 36 |
| Extract complete tables | 37 |
| Extract complete website | 37 |
| Save extracted data | 38 |
| Extract & Scripting Interface | 38 |
| Extract Tech Tip | 40 |
| Asian Language Support | 41 |
| 8 Winclick | 42 |
| Trigger Mouse Over Events | 43 |
| 9 Response Time Measurements | 44 |
| Automating Response Time Measurements | 45 |
| Tips for Accurate Web Response Time Measurements | 45 |
| 10 Change User Agent | 46 |
| 11 Filter | 47 |
| 12 Send Email | 47 |
| 13 Error Handling | 48 |
| Error Codes | 48 |
| Part IV Automation | 49 |
| 1 Shortcuts | 49 |
| 2 Batch Files | 50 |
| 3 Schedule Tasks | 52 |
| 4 Control via Scripting Interface | 53 |
| Example Windows Scripting Host | 54 |
| Scripting Example Visual Basic.NET | 55 |
| Intellisense Support..... | 55 |
| Run iMacros under a different user account | 57 |
| Start as Windows Service | 57 |
| Start from Web Page | 58 |
| ASP/ASP.NET/PHP | 59 |
| Part V Image Recognition Plugin | 60 |
| 1 Search and Click Images/Buttons | 60 |
| 2 Create Images for IMAGESEARCH | 61 |
| 3 Example: Automate Flash Chat Web Applet | 64 |
| Part VI Distributing iMacros with your Application | 65 |
| 1 Setup Command Line Parameters | 66 |
| 2 Restricted User Accounts | 67 |
| 3 Modify settings directly | 67 |
| Part VII Frequently Asked Questions | 68 |

| | |
|---|-----------|
| 1 Getting started | 68 |
| The web page I am accessing requires IE. Is this a problem? | 68 |
| Are there conditional statements in the Internet Macros language? | 68 |
| 2 Installation | 69 |
| I create macros for my clients. Do you have a free player? | 69 |
| How can I automatically install iMacros? | 69 |
| What if the iMacros icon does not appear in the IE toolbar? | 69 |
| 3 Creating Macros | 70 |
| How to... | 70 |
| How to automate pages where links (URLs) change every time I visit the page? | 70 |
| How can I print a selected frame? | 70 |
| How do I set the focus to a input field for manual entry? | 71 |
| How to create macros that will run on the page displayed in the web browser? | 71 |
| How to read and write from a database? | 71 |
| How do I make the macro continue (and not stop), if somewhere in the macro I get a timeout or error? | 71 |
| How can I do calculations in a macro? | 71 |
| How do I link several macros together? | 72 |
| How to make iMacros stop until a users enters a value? | 72 |
| How to create a macro that can select one from a series of radio buttons? | 72 |
| How to display the content of a variable? | 73 |
| Issues during Replay | 73 |
| Does iMacros work with every web site? | 73 |
| Can we have a loop inside a macro? | 73 |
| Does the macro script wait for the page to fully finish loading? | 74 |
| Why is a certain input box never recorded? | 74 |
| Why is dialog XYZ not handled by iMacros? | 75 |
| How do I make the iMacros Browser appear as native IE (Internet Explorer)? | 75 |
| 4 Extracting Data | 75 |
| I use the EXTRACT command and get the message "Extraction anchor not found". What is the solution? | 75 |
| How can I insert the extracted information back into same webpage? | 76 |
| How to extract information from a table with variable length and/or more than one page? | 76 |
| How to work (fill/extract) with hidden input fields? | 78 |
| How do I extract text from message boxes? | 78 |
| How do I extract data separated with ? | 79 |
| How do I extract the page URL? | 79 |
| 5 Web Testing | 79 |
| How can I search for a specific keyword on a web page? | 79 |
| Keyword Search with TAG | 79 |
| Keyword Search with EXTRACT | 80 |
| Keyword Search with IMAGESEARCH | 81 |
| How to test that certain images will show up when a page is loaded? | 81 |
| What effects has the iMacros Browser itself on application response measurements? | 82 |
| How to set up a 7x24h (non-stop) operation? | 82 |

Part VIII iMacros Commands Reference 83

| | |
|----------------------|-----------|
| 1 ADD | 87 |
| 2 BACK | 88 |
| 3 CLEAR | 88 |
| 4 CLICK | 89 |

| | | |
|-----------------------------------|---------------------------|------------|
| 5 | CMDLINE | 90 |
| 6 | comments | 91 |
| 7 | DISCONNECT | 91 |
| 8 | EXTRACT | 91 |
| 9 | FILEDELETE | 93 |
| 10 | FILTER | 94 |
| 11 | FRAME | 94 |
| 12 | IMAGECLICK | 95 |
| 13 | IMAGESEARCH | 96 |
| 14 | ONCERTIFICATEDIALOG | 97 |
| 15 | ONDIALOG | 97 |
| 16 | ONDOWNLOAD | 98 |
| 17 | ONERRORDIALOG | 99 |
| 18 | ONLOGIN | 99 |
| 19 | ONPRINT | 100 |
| 20 | ONSEcurityDIALOG | 101 |
| 21 | ONWEBPAGEDIALOG | 101 |
| 22 | PAUSE | 102 |
| 23 | PRINT | 103 |
| 24 | PROMPT | 103 |
| 25 | PROXY | 104 |
| 26 | REDIAL | 105 |
| 27 | REFRESH | 106 |
| 28 | SAVEAS | 106 |
| 29 | SET | 107 |
| 30 | SIZE | 108 |
| 31 | STOPWATCH | 109 |
| 32 | TAB | 110 |
| 33 | TAG | 110 |
| 34 | URL | 113 |
| 35 | VERSION | 114 |
| 36 | WAIT | 114 |
| 37 | WINCLICK | 115 |
| Part IX Built-in Variables | | 116 |
| 1 | !COLn | 119 |
| 2 | !DATASOURCE | 119 |
| 3 | !DATASOURCE_COLUMNS | 120 |
| 4 | !DATASOURCE_LINE | 120 |
| 5 | !DIALOGMANAGER | 120 |

| | | |
|-------------------------------------|---------------------------|------------|
| 6 | !ENCRYPTION | 121 |
| 7 | !ERRORIGNORE | 121 |
| 8 | !ERRORMACRO | 122 |
| 9 | !EXTRACT | 122 |
| 10 | !EXTRACT_TEST_POPUP | 123 |
| 11 | !EXTRACTADD | 123 |
| 12 | !EXTRACTDIALOG | 123 |
| 13 | !FILELOG | 124 |
| 14 | !FILESTOPWATCH | 124 |
| 15 | !FOLDERIMACROS | 124 |
| 16 | !IMAGEX | 125 |
| 17 | !IMAGEY | 125 |
| 18 | !LOADCHECK | 125 |
| 19 | !LOOP | 126 |
| 20 | !MACROTIMEOUT | 126 |
| 21 | !NOW | 127 |
| 22 | !POINTER | 129 |
| 23 | !REPLAYSPEED | 129 |
| 24 | !STOPWATCHTIME | 130 |
| 25 | !TIMEOUT | 130 |
| 26 | !URLCURRENT | 130 |
| 27 | !URLSTART | 131 |
| 28 | !VAR1 | 131 |
| 29 | !VAR2 | 131 |
| 30 | !VAR3 | 132 |
| 31 | !VARDEFAULT | 132 |
| Part X Command Line Switches | | 132 |
| 1 | datasource | 134 |
| 2 | loop | 134 |
| 3 | macro | 135 |
| 4 | noexit | 135 |
| 5 | silent | 135 |
| 6 | timeout | 136 |
| 7 | tray | 136 |
| 8 | useragent | 137 |
| 9 | var_varname | 137 |
| 10 | var1 | 137 |
| 11 | var2 | 138 |
| 12 | var3 | 138 |

| | |
|---|------------|
| Part XI Scripting Interface Command Overview | 139 |
| 1 iimDisplay | 139 |
| 2 iimExit | 140 |
| 3 iimGetLastError | 141 |
| 4 iimGetLastExtract | 141 |
| 5 iimInit | 142 |
| 6 iimPlay | 143 |
| 7 iimSet | 144 |
| 8 Scripting Interface Return Codes | 145 |
| Part XII How to buy iMacros | 146 |
| Part XIII Feature comparison | 147 |
| Index | 148 |

1 Introducing iMacros

What is iMacros™ ?

In a nutshell, it is the world's first browser-based macro recorder. It allows you to easily record web surfing and replay it.

The web browser is probably the most frequently used software today, but many tasks are repetitious: checking on the same sites everyday, remembering passwords, submitting to search engines or testing web sites over and over again. With iMacros, you record these tasks once and then let iMacros execute them whenever you need them.

Any combination of browsing, form filling, clicking and information gathering can be recorded into a macro and iMacros even assists you during the recording with visual feedback.

- **Do you need to extract price lists, stock information or any other data from websites?**

iMacros can do this for you. It submits data from a file to a website and stores the result from the website in a text file. No programming skills required!

- **Do you need to test web sites automatically?**

Have you ever spent hours browsing your website to re-test it after a change? Our advanced PRO and Scripting Editions can automate almost any kind of web regression and verification testing for you!


- **How does iMacros Scripting Edition compare to big name capture/replay web test software on the market?**

iMacros Scripting Edition successfully competes with website testing software priced more than US\$ 30,000 for only a fraction of the cost! In addition, the Scripting Edition allows you to interface iMacros with the well-documented Windows Scripting Host (included in Windows), Visual Basic or any other programming language that runs under Windows. This is a crucial feature for fully automating any given task and is often missing in much more expensive software.

What can iMacros™ do for you?

The following list are only some examples of many possibilities. It can do (almost) everything that you can do with a web browser!

If you use the Internet Explorer Plug-In (Power User Edition), it can do the following:

- Automatically complete forms.
- Auto-login to your webmail (and, if you want, also write and send the email for you!).
- Navigate complex websites repeatedly without user intervention.
- Transparent, transportable macro files: record on one PC, replay on any other PC that has iMacros installed. In contrast, Microsoft's AutoComplete entries can *not* be copied from one PC to another.
-  storage of passwords using the industry standard 256-bit AES encryption algorithm. AES is also used by U.S. Government organizations to protect sensitive information.
- Keep your privacy: unlike some other form filling tools, iMacros does *not* send any data back to us. All data is only stored on *your* local PC.
- Share your macros with your colleagues and increase productivity for your whole organization.

If you are a Web Professional and use the PRO Edition, it can do the following:

- Regression test whole areas of complex web sites at the click of a button.
- Automate your search engine submissions, manage Google and Overture Pay per Click search

engine listings and biddings automatically, extract information from web sites, query online databases and download the results automatically.

- Internet Monitoring: watch your web site and alert you if the macro encountered a problem on your web site. In contrast to plain Internet Monitoring services, iMacros can test online forms of any complexity (e.g. create test orders in an Online store) including Java and Macromedia Flash based elements.
- Measure web site response times with the `STOPWATCH` 109 command to create performance statistics.
- Avoid complicated Perl scripts, Cron jobs, grep, sed, awk, lwp and other time-consuming Unix tools and commands. In addition, none of these tools have the wide functionality that iMacros has!
- iMacros can be set to simulate Internet Explorer (IE) completely. In this mode, it is *not* possible at all for a web server to distinguish between a normal (human) user and the iMacros robot.
- Extract any table directly into a comma-separated text (CSV) file. These files can be read by almost any spreadsheet software, including Excel.
- iMacros supports web sites based on Java applets or Flash (the Macromedia Shockwave plug-in).
- Supports XHTML.

If you are a Web Professional or a software developer and use the Scripting Edition, it can do the following:

- Extract data from web pages (web queries).
- Web enable your application in 5 minutes.
- You just have to write a macro and call the iMacros command line or Scripting interface. iMacros does the rest!
- Use iMacros as an Internet agent, robot or spider.
- Add "web surfing" and "web query" capability to your Windows Scripts.
- Ship iMacros with your application. The Scripting Edition comes with a special redistribution license *without* royalty fees.

You don't have time to create iMacros yourself? You need more complicated internet functions automated? We create customized solutions for you based on our award-winning innovative software. Please ask or a free quote!

System Requirements

Windows 98, ME, 2000, XP or Server 2003 with 486 or higher processor (Pentium recommended)
8MB of free hard drive space for installation Microsoft Internet Explorer 5.5 or higher.

iOpus, iMacros, iMacros are among trademarks of iOpus Software in the United States and other countries. All other trademarks and tradenames belong to their respective owners.

1.1 Why iMacros?



Save time

iOpus iMacros helps you perform your web chores quicker. Downloading, data entry and web site testing - iMacros can do all that for you!



Save money

Why pay more for less? iMacros is THE low cost web testing solution and it even outsmarts its US competitors in many features. Some of the competition charges as much as \$30,000 and still has less capabilities than iMacros!



Easy

You will create your first Internet Macro in less than a minute! No other web automation software is that easy to use.



Flexible

Automate even the most complicated tasks with the Scripting Interface. Connect iMacros to your favorite programming language. Windows Scripting Host and Visual Basic example programs are included.



Document Web page changes

iMacros can save web pages or even print them out directly.



Be creative

Repetition is unavoidable, but you avoid almost all of it. Let iOpus iMacros take over the routine jobs, and save your precious time for the creative part.

2 Using iMacros

2.1 Start iMacros

iMacros can be used in two different modes:

Internet Explorer Plug-in:

After installing the software, a new icon called "iOpus iMacros" appears in the menu bar of Internet Explorer (IE). Click this icon to start iOpus iMacros. If you have already customized your toolbar, you might need to add the icon to the Internet Explorer toolbar [manually](#)^[69].

The special Internet Explorer Browser [PRO and SCRIPTING Edition]:

To start this program, click on the icon on your desktop, or navigate to the file "imacros.exe" and click on it.

To become familiar with the iMacros, [run](#)^[12] the "Demo-FillForm" macro that is automatically installed. It demonstrates the basic features of iMacros in an uncluttered way. It navigates to a test form, fills the form, sends it and returns to the iOpus Internet Macro Website.

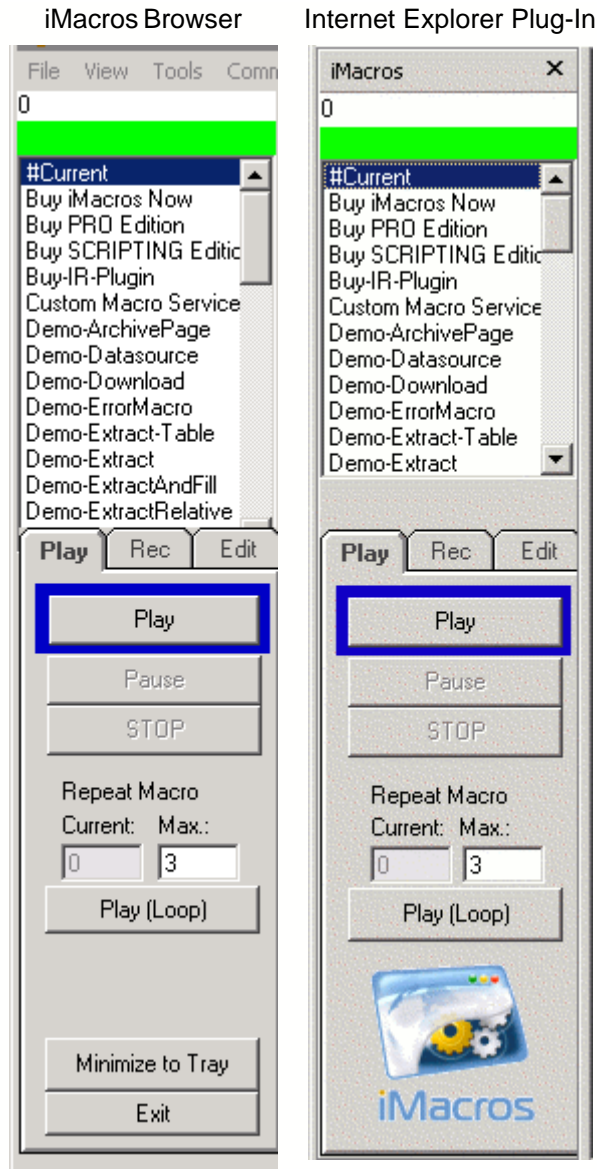
2.2 The iMacro Graphical User Interface

We have invested a lot of effort in making the Graphical User Interface of the iMacros Browser and the Internet Explorer Plug-In as intuitive as possible. In this section the basic elements are explained.

There are minor differences between the iMacros Browser and the Plug-In, which will be pointed out as necessary.

The main window


The main window consists of two parts, on the left you see the iMacros panel and on the right you see the browser window. The website is displayed in the browser window just as you are used to when using any other browser. With the elements on the left you control iMacros.


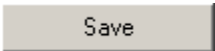


The iMacros control can be divided in two parts. At the top, you see all your macros. Macros are only recognized by iMacros if they have the file ending .iim and are lying in the Macros directory of your iMacros installation. The default for this is C:\Program Files\iMacros\Macros\.

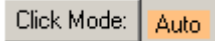
The bottom part consists of three tabs, Play, Rec and Edit. As the name suggests, in the Play tab you will find controls for playing macros, in the Rec tab controls for recording and in the Edit tab you will find controls for editing.

2.3 Recording

To record your own macros, select the Record tab of iOpus iMacros. Click  to start a recording. Now simply perform whatever tasks you wish to record, such as surfing to web sites, filling out forms, making a test order on your e-commerce site or any other task of your choosing.

After you have finished recording, press . The current recording is automatically saved as #Current macro. Press  to save it under an individual name. Upon saving, the macro is parsed and iMacros attempts to remove all double or multiple entries.

Hints for successful recording:


- A **GREEN** frame around a tag means that iOpus iMacros supports this tag.
 - A **RED** frame around a tag means that you have selected a web page element that is not useful for recording such as BODY, which has no active functions.
 - During recording, do *not* use your browser's Back or Forward button or change the address line. Use the functions supplied by iOpus iMacros instead. [Internet Explorer Plug-in only]
 - Recording "Clicks":
If a macro generates an error during replay, you can experiment by selecting the link recording option (Click Mode) manually from the dialog after clicking .
1. **Automatic:** iMacros attempts to choose the best recording options (recommended in most cases).
 2. **Use Link Name:** The link is identified by its name. This works well in most cases, except when there are several links with the same name on a page.
 3. **Use Link URL:** The link is detected by its URL. If the URL changes each time you visit a page, this option is not recommended unless you replace the changing part of the URL with an *. Here is [more information](#)^[24].
 4. **Use X/Y:** The link is identified by its position on the web page. This can be useful if the name and URL of the item you want to click on changes dynamically from visit to visit. It should also be used if the web page element is embedded in JavaScript, so that iMacros can not find the recorded name during replay, because it was created "on the fly" by JavaScript.
 5. **Use Windows Clicks**^[42]: This is the ultimate solution if everything else fails. It simulates standard mouse clicks within the browser window. This feature works with *all* Web pages but is only available in the iMacros Browser.
- Because the iMacros recording language is fully [documented](#)^[83], it is easy to edit and "tweak" the macro manually after recording!
 - iMacros supports two recording options: FAST (the default) and ORIGINAL SPEED. This can be set in the Settings tab of the Options dialog. If you record in the original speed mode, iOpus iMacros automatically generates [WAIT](#)^[114] statements so that the replay is at the same speed as your recording. This is very useful for recording demos. For most purposes, however, you want to replay the tasks as fast as possible.

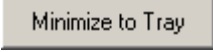
Tip: Be sure that you check our support page at <http://www.iOpus.com/iim-support.htm> for the latest recording tips & tricks or submit any recording problems at our online support form at

<http://www.iOpus.com/service/support.htm>.

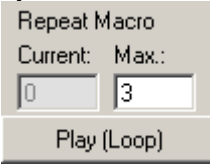
2.4 Replay

To replay any macro from the macro directory, open the Play tab, select a macro from the list and

press . During replay a blue frame shows you which parts of the web site are

being manipulated. To run iMacros in the background, click on the  button [iMacros Browser only].

To replay the recorded sequence several times, fill in the number of loops and press Play

 (Loop):

There are two different options that affect the speed at which macros are replayed. These can be set in the Settings tab of the Options dialog or they can also be changed within the macro with the [!REPLAYSPEED](#)^[129] variable.

The first option is the replay speed, which can be set to three different values:

- FAST: The macro is replayed at maximum speed (Recommended in most cases).
- MEDIUM: iMacros waits for 0.25s between each command.
- SLOW: iMacros waits 1s between each command.

The second option is to insert [WAIT](#)^[114] statements during [recording](#)^[11]. If the checkbox marked "RECORD original speed" is marked, [WAIT](#)^[114] statements are included. Thus, during replay, these [WAIT](#)^[114] statements slow down the process of replaying.

Tip: If you do want the blue frame to appear during replay, add the following statement to your macro:

```
SET !POINTER[129] YES
```

Errors during replay

We work hard to make iMacros as "intelligent" as possible, but it still is not as smart as you. If an error occurs during replay, it is mostly due to a "tricky" web page at which one of the automatic suggestions of the iMacros Recorder failed. In almost all cases, re-recording the macro with different settings or manually editing the macro solves the problem. For recording tips, please see [Recording](#)^[11] and for information on how to edit your macro, go to the [Edit Macros](#)^[12] section.

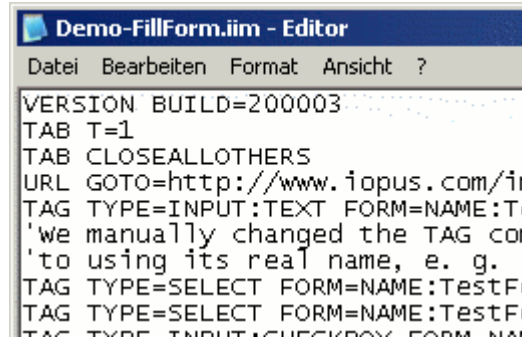
2.5 Edit Macros

Related example macros: *Demo-FillForm*

All recorded macros are stored in a plain text file with the ending `.im` in the directory defined by the Folder Macros text field in the Path tab of the Options dialog, e.g. `C:\Program Files\iMacros\Macros\`. You can manually edit and tweak the macros using any text editor you like, e.g. Notepad, which is shipped with the Windows operating system by default. To open any macro, open the Edit tab, mark the macro you want to edit and click the

Edit Macro

button. The editor you entered in the Path Macro Editor text field located in the Path tab of Options dialog (Notepad by default) will open and display the macro - in this example we chose Demo-FillForm:



You could now change the content of a form text field. To change the Name below from "Tom Tester" to "Dr. A. Award", locate the [TAG](#)^[110] command that contains "Tom Tester" and change is as shown below:

Old: TAG POS=1 TYPE=INPUT:TEXT FORM=NAME:f1 ATTR=NAME:n1 CONTENT=Tom Tester

New: TAG POS=1 TYPE=INPUT:TEXT FORM=NAME:f1 ATTR=NAME:n1
CONTENT=Dr.<SP>A.<SP>Award

When assigning values to the CONTENT parameter of any iMacro command, all whitespaces in the text must be substituted by <SP> and all newlines must be substituted by
.

After you saved the changes to the file, iMacros will immediately apply them during the next replay of the macro.

2.6 Security

Many web sites require you to type a user name and password before you can enter the site. For instance, personalized pages and web sites containing your financial information require you to log in.

The iMacros Password Manager can help you by storing your user names and passwords in macros, and entering them for you automatically when you visit such sites.

There are three ways to store passwords in macros you record. The password encryption method can be either set or disabled for all macros in the Security tab of the Options dialog or individually for each macro with the [!ENCRYPTION](#)^[121] variable.

1. No encryption

The password is stored inside the macro in plain text. This method is very convenient, but keep in mind that everybody who opens the macro can read the password.

2. Encrypted web site passwords

Passwords are encrypted using a strong 256-bit encryption based on the industry-standard AES algorithm. This encryption requires a master password, which is stored on your own computer in the iMacros settings file. The default master password is "iOpus2004". This master password can and indeed should be changed in the Security tab of the Options dialog. The iMacros settings file is very difficult, but not impossible, for an intruder to read. For macros that need to run unattended, this is the


best solution possible as every automatic solution needs to store the password somewhere. This is the default setting.

3. Encrypted web site passwords and ask for the Master Password

Passwords are encrypted using the same strong 256-bit encryption based on the industry-standard AES algorithm as in 2., but the master password is not stored. It is only kept *temporarily* in memory while you run the macros. You need to re-enter it once when you start iMacros and use a website password the first time; much more secure than the other two options, but less convenient. This means that even if somebody steals your PC, they can not run the macros which include website access using passwords. This method is recommended for macros that you start manually, such as your personal online banking macros.

2.7 Saving Web Sites

Related example macro: *Demo-ArchivePage, Demo-SaveAs*

iMacros automatically downloads and saves web pages for you. Use the  button and select the format from the iMacros Save As dialog. iMacros will then insert a [SAVEAS](#)^[106] command in the macro, holding information about the format and the location where the file is saved. The default location is in the `downloads\` directory of your iMacros installation. This command is very easily tweaked after the macro has been recorded to fit your needs.

You have different options as to which format you wish use to save the currently displayed web page. These options are

CPL

The complete web page is saved. The files and images are saved separately and stored in a folder.

MHT

The web page plus images are saved in a single file (Web Archive).

HTM

The web page source is saved with no images. If the page has frames, all framed HTML pages are saved automatically.

TXT

Only the web page text is saved; all HTML tags are omitted.

EXTRACT

The value of the variable [!EXTRACT](#)^[122] is saved in CSV format.

BMP

A [screenshot](#)^[14] of the web page is saved.

2.8 Web Site Screenshot

Related example macro: *Demo-TakeScreenshot*

The iMacros Browser can automatically take screenshots of web pages. This includes the part "below the fold", i.e. iMacros takes the screenshot of the entire web page, no matter the length, even if it scrolls off-screen!

To take a web page screenshot, insert a [SAVEAS TYPE=BMP](#)^[106] command into the macro like in this


```
SAVEAS TYPE=BMP FOLDER=* FILE=My_Screenshot.bmp
```

The file format is the standard Windows Bitmap format (BMP). This file format can be used with any image editor. Also, it can be compressed effectively with any [ZIP compression tool](#) or the Windows built-in "[compress contents](#)" option for folders.

[illegible]

© 2001 - 2006 iOpus Software GmbH. all rights reserved.

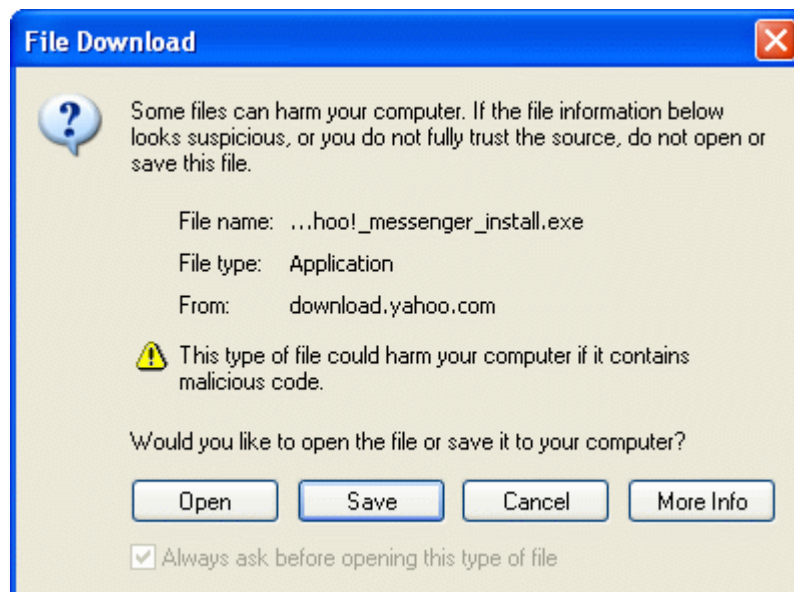
2.9 Save Web Site Elements

iMacros can intercept file downloads and even download pictures and other elements from the web site you are visiting.

2.9.1 Download Files

Related example macro: *Demo-Download*

Whenever a file download is initiated, Internet Explorer will present you with the following dialog. This dialog allows you to select what you would like to do with the file. If you choose to save it, it will also ask you for the file name and location.



Downloads are controlled by the [ONDOWNLOAD](#)^[98] command. It has two parameters that control the file name and the folder in which to save the file. It must appear before the macro command that starts the download.

Note: The general web page timeout also applies to downloads. So make sure that the timeout value is long enough to cover the complete download time. If needed, you can also increase the timeout value manually inside the macro with the [SET](#)^[107] [!TIMEOUT](#)^[130] command. Alternatively, you can add a [WAIT](#)^[114] SECONDS=#DOWNLOADCOMPLETE# command.

2.9.2 Save Item

Related example macro: *Demo-ImageDownload*

To download an image (or any other specific web page element) switch to [recording](#)^[11] mode, navigate to the page and press [Save Item](#) in the iMacros Browser or [Save Item as](#) in the Internet Explorer Plug-In.

The item download preview opens in the tab on the left. Next, select the image or item in the web browser window. The selection is then displayed in the preview section. If you like the selection, press ok. The item download manager automatically adds a [TAG](#)^[110] line to your macro, e.g.:

```
TAG POS=1 TYPE=IMG ATTR=HREF:http://www.iopus.com/logo.jpg
CONTENT=EVENT:SAVEITEM
```

You can also create download tag lines manually: During [recording](#)^[11], simply click on the wanted item and add `CONTENT=EVENT:SAVEITEM` to the generated [TAG](#)^[110] line in an editor of your choice.

The downloaded files are copied to the iMacros download directory (usually `C:\Program Files\iMacros\downloads\`). The items are taken directly from the web browser that displayed them. They are *not* downloaded again to save time and bandwidth. The naming convention for downloaded images is the same as for [downloaded files](#)^[16], i.e. they consists of the original file name and a date and time tag. This also means you can use the [ONDOWNLOAD](#)^[98] command to specify your name for the downloaded image.

You can also download images using [SAVEPICTUREAS](#)^[17]. The command is named after the corresponding functions in the Internet Explorer menu, "Save Picture as". You can right-click on any web page element to see if the web page element supports this feature; if it does, iMacros can handle it automatically for you. To create such a command, click on the element you would like to save during record and add `CONTENT=EVENT:#SAVEPICTUREAS` manually to the generated [TAG](#)^[110] command. To control the location and name of the downloaded file, use the [ONDOWNLOAD](#)^[98] command.

2.9.3 Save Target As

Related example macro: *Demo-SaveTargetAs*

To download any kind of web content such as .WVM or .AVI videos, sound files such as .MP3, or documents such as .PDF, first click on the link that connects to the item. In the case of a .PDF file, such a link is typically called "Open document" or for a video file, "Show video". iMacros records a [TAG](#)^[110] command:

```
TAG POS=1 TYPE=B ATTR=TXT:Open<SP>PDF<SP>Document
```

This would simulate a mouse click on the link. This is not what we want, we need a command similar to the right-click command "Save Target As" in Internet Explorer. This can be achieved by editing the macro and adding `CONTENT=EVENT:SAVETARGETAS` to the recorded normal TAG command. You will then have:

```
TAG POS=1 TYPE=B ATTR=TXT:Open<SP>PDF<SP>Document
CONTENT=EVENT:SAVETARGETAS
```

The files downloaded in this way are copied to the iMacros download directory by default. You can use the [ONDOWNLOAD](#)^[98] command to specify the name and location for the downloaded image.

2.9.4 Save Picture As

Related example macro: *Demo-ImageDownload*

This command is similar to [SAVEITEM](#)^[16], which is the default command for image downloading. However, unlike `SAVEITEM`, it does not access the image via the HTML of the website but directly by simulating the Internet Explorer "Save Picture As" command. This can be an advantage on complex websites or websites where images are constructed "on the fly".

To use `SAVEPICTUREAS`, you need to manually edit a recorded macro. Click on the element you wish to download and iMacros will produce a command like

```
TAG POS=1 TYPE=IMG ATTR=HREF:http://www.iopus.com/
```

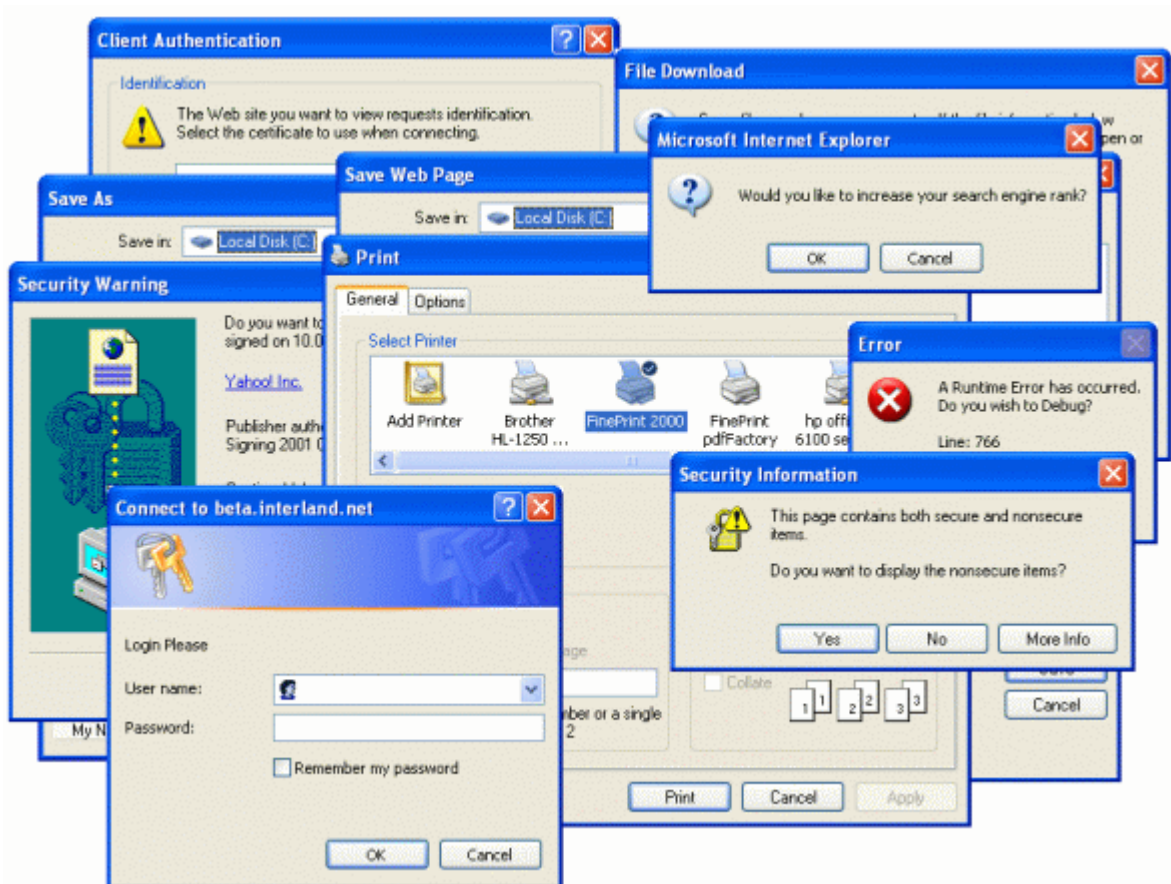
This would simulate a mouse click on the link. Since this is not what we want, we need to edit the macro and add `CONTENT=EVENT:SAVETARGETAS` to the recorded TAG command. You will then have

```
TAG POS=1 TYPE=IMG ATTR=HREF:http://www.iopus.com/
CONTENT=EVENT:SAVEPICTUREAS
```

The files downloaded in this way are copied to the iMacros download directory by default. You can use the `ONDOWNLOAD`^[98] command to specify the name and location for the downloaded image.

2.10 Dialog Manager

The Dialog Manager allows you to manage all those pesky dialogs that appear with Internet Explorer from time to time. Because the iMacros Browser emulates Internet Explorer, the same dialogs appear in the iMacros Browser too.



Sample of some dialogs that can show up while you browse the web. iMacros can handle them all!

2.10.1 Login

iMacros fills all login dialogs for you using the `ONLOGIN`^[99] command. The password is stored with the method you selected on the Security tab of the Options dialog. More details about the different

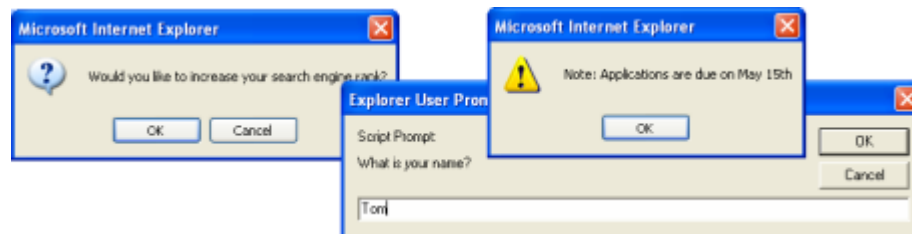
password storage options can be found [here](#)^[13].



Login Dialog Sample

2.10.2 Javascript

Related example macro: *Demo-OnJavascriptDialog*



iMacros handles all Javascript dialog boxes for you using the [ONDIALOG](#)^[97] command. You can [extract](#)^[30] the text of a dialog by adding SET [!EXTRACTDIALOG](#)^[123] YES to your macro.

Tip: On some pages, a new page loads once a button on the dialog is clicked. If you want iMacros to wait for this page to load before continuing, please add [WAIT](#)^[114] SECONDS=#DOWNLOADCOMPLETE# after the TAG statement that triggers the dialog box.

2.10.3 Web Page Dialogs

Related example macro: *Demo-OnWebPageDialog*

Web page dialogs are similar to Javascript dialogs, except they display HTML content. iMacros can control them using the [ONWEBPAGEDIALOG](#)^[101] command. Since web page dialogs can contain any number of buttons or boxes, you can automate them by sending a specific list of keyboard commands to them. For example KEYS=Hello{ENTER}{CLOSE} will enter the word "Hello" on the dialog, press ENTER key and then close the dialog.

During replay, ONWEBPAGEDIALOG KEYS={WAIT<sp>2}{CLOSE} is active by default to close

unwanted ad dialogs.

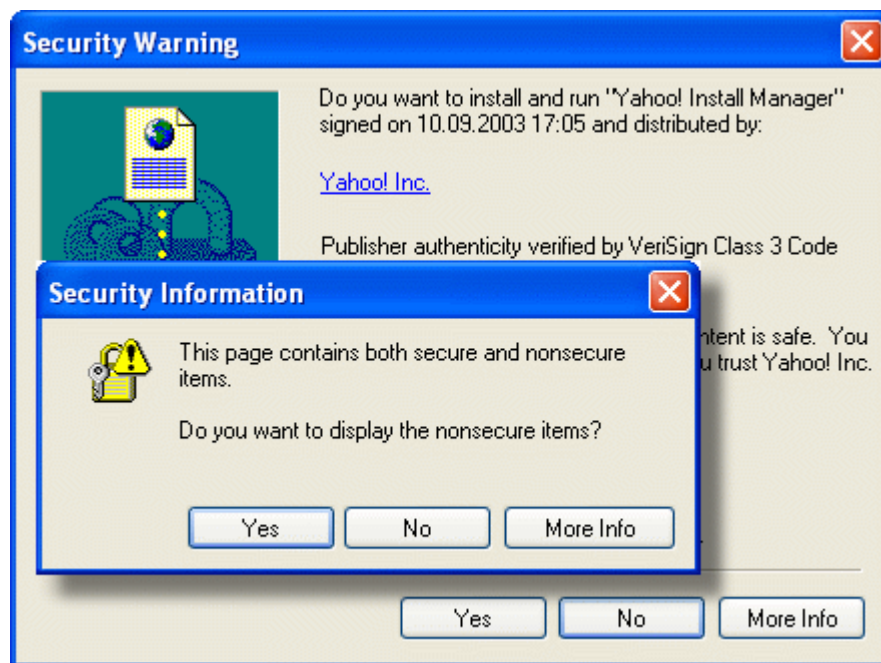
Note: Web page dialogs are not normal HTML browser windows. Therefore they do not open in a separate tab, but pop up in front of the current window.

2.10.4 Print

Related example macro: *Demo-Print*

The dialog manager works with the print dialog. For more details, please see the [PRINT command](#)^[21].

2.10.5 Security

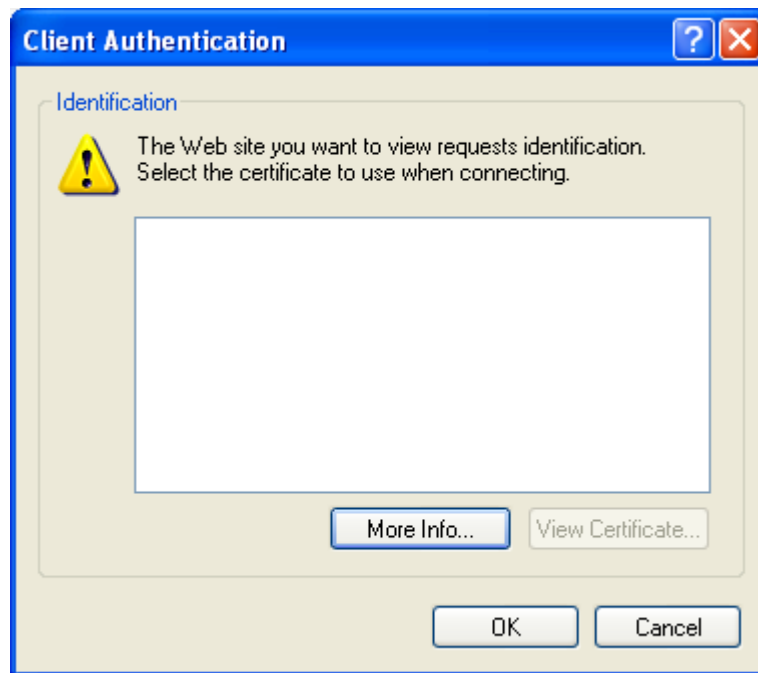


Security dialogs can occur on many secure web sites. The iMacros Dialog Manager automatically clicks the security dialog boxes, so your macros are *not* interrupted. This is done using the [ONSECURITYDIALOG](#)^[101] command.

By default, the settings are `BUTTON=YES` and `CONTINUE=YES`. These settings are active even without an `ONSECURITYDIALOG` command in your macro.

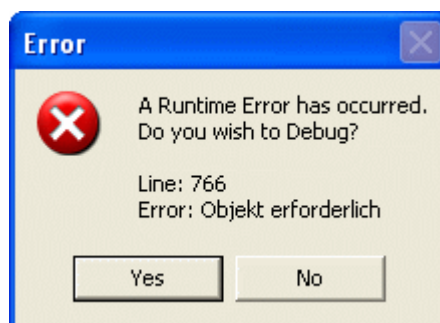
2.10.6 Certificates

Some secure web pages ask you to select a client side certificate. The dialog manager can do this for you using the [ONCERTIFICATEDIALOG](#)^[97] command.



2.10.7 Page Errors

Related example macro: *Demo-OnError*



Page script errors can occur on many web sites for a variety of reasons: The iMacros Dialog Manager automatically clicks the Internet Explorer error dialog boxes, so your macros are *not* interrupted by script errors! This is done using the [ONERRORDIALOG](#)^[99] command.

By default, the settings are `BUTTON=YES` and `CONTINUE=YES`. These settings are active even without an `ONERRORDIALOG` in your macro.

2.11 Print

Related example macro: *Demo-Print*

iMacros includes a [PRINT](#)^[103] command that triggers a print dialog. At this point, the dialog manager takes over with the values defined in the [ONPRINT](#)^[100] command. Thus you can select a specific printer by using [ONPRINT](#)^[100] `P=3`. In this case, the third printer is used. If you use only `P=` or `P=*`

the most recently selected printer is used. Typically this is the default windows printer (The only exception to this rule is if you select another printer before (e. g. with P=3) then printer #3 is the default printer during this iMacros session.).

If the page uses frames and you want to print only a specific frame, select this frame with WINCLICK^[42] first.

Example:

```
VERSION BUILD=4020412
TAB T=1
TAB CLOSEALLOthers
URL GOTO=http://www.iopus.com/iim/demo/frames.htm
SIZE X=644 Y=604
FRAME F=6
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:frame7.htm ATTR=NAME:T1
CONTENT=Print<SP>this<SP>frame<SP>only
'Make winclick to select frame
WINCLICK X=462 Y=206 CONTENT=
ONPRINT P=*
PRINT
```

2.12 Offline

If you work on a PC without Internet connection, it is recommended that you check the box "Work Offline" in the Options dialog. This avoids the Internet connection check at iMacros startup.

3 Advanced

3.1 Tabbed Browser

Related example macros: *Demo-Tab*

The iMacros Browser [PRO and Scripting Edition only] includes a tabbed browsing interface that makes managing web sites with multiple open pages a snap. When a web page opens a new window, iMacros automatically opens it in a new tab in the background. If the user changes to another tab, a TAB^[110] command is automatically added during recording.



You can close tabs while browsing by right-clicking on the tabs (not the browser window itself!). This will open up a context menu with the options to close the tabs. The following example shows the basic actions you can do with the TAB^[110] command

```
' open a webpage in the first tab
URL GOTO=http://www.iopus.com
' open a new tab
```



```
TAB OPEN
' get new tab to foreground
TAB T=2
' load another page
URL GOTO=http://www.google.com
' close the second tab
TAB CLOSE
TAB T=1
```

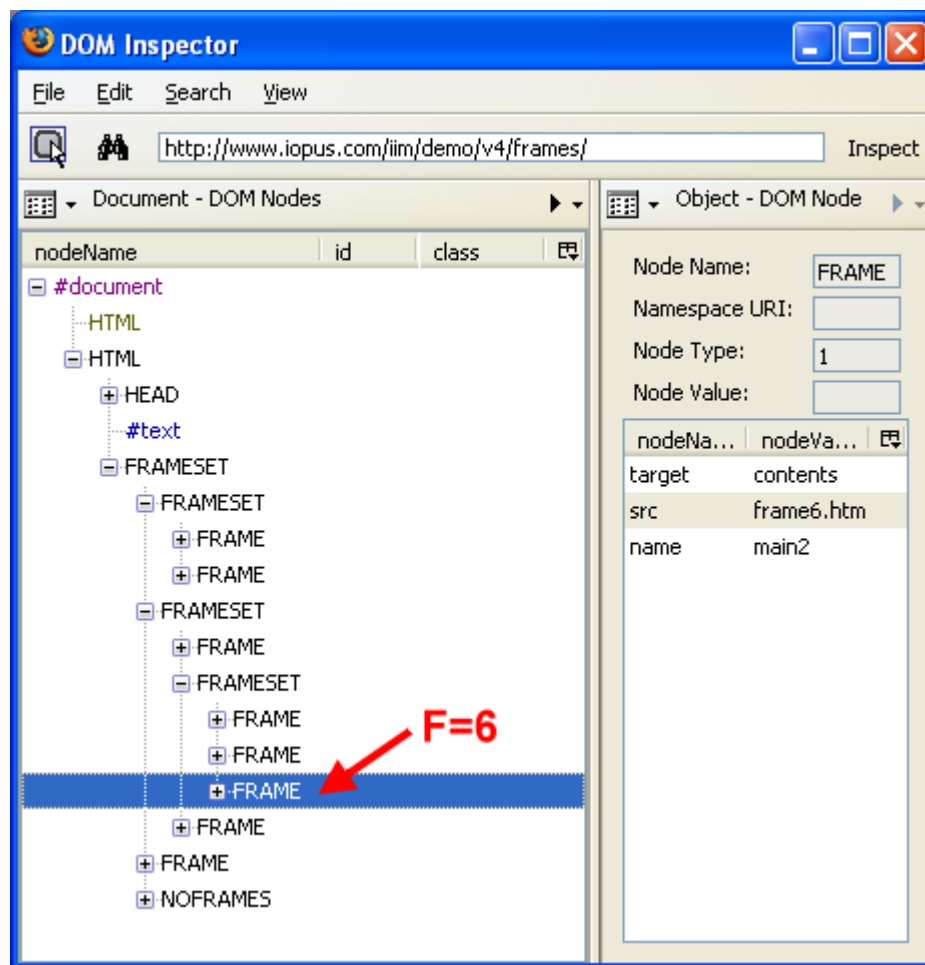
3.2 Frames

iMacros handles pages with frames automatically. It inserts [FRAME](#)^[94] statements that indicate to which frame the following [TAG](#)^[110] and/or [EXTRACT](#)^[91] command refers. Please note that [TAG](#)^[110] and/or [EXTRACT](#)^[91] will fail if they are not directed to the correct frame. To avoid this error, please do the following:

- [EXTRACT](#)^[91] command: if the object that needs to be extracted is on a framed page, make sure that you use the correct [FRAME](#)^[94] statement. You generate such a [FRAME](#)^[94] statement by clicking on any element of any kind within the frame prior to performing the extraction
- If a [TAG](#)^[110] error occurs because the pages are not completely loaded: normally iMacros waits until the browser sends a "Page Loaded" signal before it continues. On framed pages, iMacros sometimes gets confused and continues with the macro too early, i.e. before all frames have been loaded. If the content of a frame is not yet loaded, the following [TAG](#)^[110] commands will generate an error. You can resolve this problem simply by adding a [WAIT](#)^[114] SECONDS=#DOWNLOADCOMPLETE# command to your macro.

Background information

The number *n* in the [FRAME](#) F=*n* command is the position of the frame in the object tree of website:



3.3 Fine Tune TAG Commands

Normally the `TAG[110]` commands work the way they are recorded by iMacros, but sometimes you need to manually fine tune them. If an error occurs during replaying a `TAG[110]` command, it might be due to one of the following problems.

Wildcards

Some web sites are created dynamically from databases and the links contain unique numbers - the so-called session ID - each time you visit a page. While this technique helps the web site owner, it poses a problem to iMacros. This is because during recording the session ID, which is often part of links, was written into the macro as part of the `TAG[110]` command. During replay, the session ID is different, thus iMacros does not find the exact link and produces an error. The solution is to replace the changing part of a link (or extraction) with the `*` symbol, which is read by iMacros as a wildcard. The wildcard causes iMacros to except any character where the `*` is placed.

Example:

Tag line as recorded by iMacros:

```
TAG POS=1 TYPE=INPUT:TEXT
FORM=ACTION:/kb/ki.dll/ke.kb.gz?kbb;532452&&2&&&&&nc ATTR=NAME:zipcode
CONTENT=85250
```

If you record the same macro a second time, you will see that we get the same TAG line, except one

number - this is the session ID the website is using.

```
TAG POS=1 TYPE=INPUT:TEXT
FORM=ACTION:/kb/ki.dll/ke.kb.gz?kbb;532244&&2&&&&&nc ATTR=NAME:zipcode
CONTENT=85250
```

Replace the session ID with *:

```
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:/kb/ki.dll/ke.kb.gz?kbb;* &&2&&&&&nc
ATTR=NAME:zipcode CONTENT=85250
```

Actually, you could also remove most of the static parts of the FORM information as well. Exactly how much you can remove depends on the website. You still need enough information for iMacros to uniquely identify the page element. In our example, the result looks like:

```
TAG POS=1 TYPE=INPUT:TEXT FORM=ACTION:/kb/* TTR=NAME:zipcode CONTENT=85250
```

3.4 Variables

Related example macros: *Demo-Datasource, Demo-Slideshow*

Variables are, as the name suggests, constructs that allow you to dynamically, usually during runtime, hold different values. This is very helpful when you are trying to follow links that contain changing words or when you want to use the same macro for entering different values into a search engine.

The values (content) of all variables in iMacros are accessed by putting *two curly brackets* around the variable name. The values of !VAR1 is thus accessed by {{!VAR1}}.

Variables can be part of *anything* inside the macro (except the commands themselves). For example, you can add them as part of the ATTR string in a [TAG](#)^[110] or [EXTRACT](#)^[122] command or as part of the [URL](#)^[113] statement:

```
URL GOTO=https://www.onlinestore.com/?shoppingcart={{!VAR1}}&item={{!VAR2}}
```

You can assign almost any value to a variable. However, when assigning a value to a variable, certain characters need to be escaped or substituted because they imply a certain behaviour to iMacros. When assigning values to variables, all whitespaces in the value part must be substituted by <SP> and all newlines must be substituted by
; double curly brackets must be escaped with #NOVAR# ie. #NOVAR#{{.

There are two kinds of variables in iMacros:

Built-in variables:

[These variables](#)^[116] are used to define certain properties of the macro's behavior, for example the macro timeout value:

```
SET[107] !TIMEOUT 33
```

There are three special built-in variables, [!VAR1](#)^[131], [!VAR2](#)^[131] and [!VAR3](#)^[132]. These variable can be set to anything you like. They are also defined with the [SET](#)^[107] command

```
SET[107] !VAR1 hello<SP>world
```

Alternatively, you can prompt the user to input a value:

```
PROMPT[103] Please<SP>enter<SP>text !VAR1
```

User-defined Variables [PRO and Scripting Edition]

These variables are created during runtime ("on the fly") by two different mechanisms. Either one uses the command line switch^[50] -var MYVAR value^[137], like so:

```
imacros.exe -macro myMacro -var_ITEM 15
```

creates the variable `ITEM` during replay of the macro `myMacro` and gives it the value 15.

The second options is to use the iimSet^[144] function of the Scripting Interface^[53]. In a Visual Basic Script example this would look like:

```
iret = imacros.iimSet("-var_ITEM", "15")
```

3.5 Proxy Server

The PROXY^[104] command instructs the iMacros Browser to connect to the Internet through a proxy server by using the settings you specify. A proxy server acts as an intermediary between your internal network (intranet) and the Internet, retrieving files from remote Web servers. You can define a specific proxy server for each macro. Each running instances of the iMacros Browser can have its own proxy server.

If a proxy server is active, the word "PROXY:" is displayed in the status bar:



The following examples show the general use of the PROXY^[104] command. E.g., this command uses a local proxy server for both http and https at the address 192.1.8.1 and the port number 8080. Since no bypass is specified, the default settings are used.

```
PROXY ADDRESS=192.1.8.1:8080
```

This command specifies two different proxy server for the http and https protocol. Defines no bypass so iMacros uses these proxy servers even for local addresses.

```
PROXY ADDRESS=http=192.1.8.1:8080<SP>https=192.1.8.2:8080 BYPASS=NULL
```

To use a proxy server at address 66.98.229.110, but not to use it for URLs including the word "iopus", use

```
PROXY ADDRESS=66.98.229.110:8080 BYPASS=*iopus*
```

You can also use the same command, but with URL instead of IP address.

```
PROXY ADDRESS=www.iopus.com:8080 BYPASS=*iopus*
```

3.6 Submitting Multiple Datasets to Web Sites

Related example macros: *Demo-Datasource, Demo-Loop-CSV-2Web*

Related example script: *Datasource-2-web.vbs, File-2-web.vbs, File-2-web-Method2.vbs, Database-2-web.vbs*

Are you tired of filling out the same form over and over again? Then let iMacros help you. Simply put all data to be input into a very straightforward and easily understandable text file and iMacros can read the data from there and submit it to the web site - completely automatic, without your interaction!

The data source can be in either of two different formats: a text file with a list of variables and their values of the form `key=value` or as a comma separated text file (CSV format). A text file in CSV format can be generated and edited by Microsoft Excel and many other applications.

As a rule of thumb, the ["list of variables" format](#)^[28] is recommend if you have many different variables but only one or a few value(s) for each variable (for example, your detailed address data that you use to fill out online forms). The [CSV format](#)^[27] is most appropriate for use with a few variables with many different values (for example, a long list of CD's that you want to submit to an auction web site).

More advance users might [connect directly to databases](#)^[29] to retrieve the data.

3.6.1 Input from Comma Separated Data (CSV) File

Related example macros: *Demo-ReadCSV*

Related example script: *CSV-2-web.vbs, Database-2-web.vbs*

iMacros allows you to specify a text file with comma separated values to be used as input. Imagine, for example, that you want to submit a list of CD's to an online auction. Here is the list of the CD's in the comma separated format:

```
"ARTIST" , "ALBUM TITLE" , "PRICE"
"Beatles" , "Abbey Road" , "13.49"
"Beatles" , "The Beatles 1,2,3" , "25.49"
"Mozart" , "Symphonies No.40 & 41" , "9.98"
"Mozart" , "Requiem" , "7.50"
```

Note: Quotation marks are optional in most cases. They are only required if the value itself contains a comma.

We now need to tell the iMacros macro where the data input file can be found. For that, we use the built-in variable [!DATASOURCE](#)^[119]

```
SET !DATASOURCE OnlineAuction.csv
```

If you do not use any path information (like `C:\myPath\`) in the `!DATASOURCE` value, the file is assumed to lie in the standard datasources directory, which can be specified in the Paths tab of the Options dialog. The default directory is in the `datasources\` directory of your iMacros installation (e.g. `C:\Program Files\iMacros\datasources\`).

We then need to tell iMacros how many columns the CSV file has in each line. We do that by using the [!DATASOURCE_COLUMNS](#)^[120] variable:

```
SET !DATASOURCE_COLUMNS 3
```

This number *must* match the exact number of columns in the input file, even if you do not use some

columns.

Since we want to insert all datasets into the form, we need to loop over the macro, each time inserting the next CD. Therefore, we need to tell iMacros in which line of the datasource we currently are. We do this using the built-in variable `!DATASOURCE_LINE`^[120]. By cunningly using the built-in variable `!LOOP`^[126], we let iMacros take care of the counting:

```
SET !DATASOURCE_LINE {{!LOOP}}
```

Now we can have the macro fill out the online form with the values from the current CD dataset. We use the built-in variables `!COLn`^[119], where *n* represents the number of the columns to put into the form element.

```
TAG TYPE=INPUT:TEXT FORM=Listing ATTR=NAME:Name CONTENT={{!COL1}}
TAG TYPE=INPUT:TEXT FORM=Listing ATTR=NAME:Album CONTENT={{!COL2}}
TAG TYPE=INPUT:TEXT FORM=Listing ATTR=NAME:Price CONTENT={{!COL3}}
```

During the execution of the macro, the constants in parentheses `{{ . . }}` are replaced by the value specified in the data sources.

3.6.2 Input from List of Variables File

Related example macros: *Demo-Datasource*

Related example script: *Datasource-2-web.vbs*

You can also use data from a list of variable input file. The information in this format is saved in the form of key-value pairs, like

```
key1=value1
key2=value2
```

To use input in that format, create a plain text file that contains the information you want to submit. The first line in this input file must be

```
[iOpus]
```

In this example we create a file that contains the information on ordering "lunch". `lunch.txt` could look like this:

```
[iOpus]
name=Mr.<SP>Tester
main=2
drink=2
smallsize=NO
myremarks=Deliver<SP>to<SP>home<SP>address:<BR>1629<SP>4th<SP>Avenue<BR>Thanks!<BR>Tom
```

We now need to tell the iMacros macro where the data input file can be found. For that, we use the built-in variable `!DATASOURCE`^[119]

```
SET !DATASOURCE lunch.txt
```

If you do not use any path information (like `C:\myPath\`) in the `!DATASOURCE` value, the file is assumed to lie in the standard datasources directory, which can be specified in the Paths tab of the Options dialog. The default directory is in the `datasources\` directory of your iMacros installation

(e.g. C:\Program Files\iMacros\datasources\).

After we have specified the location of the input data, it is already ready to use! iMacros creates variables called after the key part of the key-value pairs. The value of this variable is the value of the key-value pair. You can now use these variable to fill the form:

```
TAG TYPE=INPUT:TEXT FORM=NAME:TestForm2 ATTR=NAME:Name CONTENT={{name}}
TAG TYPE=SELECT FORM=NAME:TestForm2 ATTR=NAME:main CONTENT={{main}}
TAG TYPE=SELECT FORM=NAME:TestForm2 ATTR=NAME:drink CONTENT={{drink}}
TAG TYPE=INPUT:CHECKBOX FORM=NAME:TestForm2 ATTR=NAME:C8&&VALUE:ON
CONTENT={{small sizedrink}}
TAG TYPE=TEXTAREA FORM=NAME:TestForm2 ATTR=NAME:Remarks
CONTENT={{myremarks}}
```

During the execution of the macro, the constants in parentheses {{ . . }} are replaced by the value specified in the data sources.

Multiple Sets of Key-Value Pairs

If you want iMacros to read different sets of key-value pairs for each loop, add a number at the end of the key and add `!LOOP`^[126] to the end of the name inside the macro. The datasource `input.txt` could then look like this:

```
[iOpus]
name1=Tom<SP>Tester
name2=Ann<SP>Smith
name3=Nicole<SP>Frank
main1=2
main2=1
main3=3
```

The complete macro would then look like this:

```
SET !DATASOURCE input.txt
TAG TYPE=INPUT:TEXT FORM=NAME:TestForm2 ATTR=NAME:Name
CONTENT={{name!LOOP}}
TAG TYPE=SELECT FORM=NAME:TestForm2 ATTR=NAME:main CONTENT={{main!LOOP}}
```

3.6.3 Input from Database

Related example macros: *Wsh-Submit-2-Web*

Related example script: *File-2-web-Method2.vbs, Database-2-web.vbs*

This example only works with the Scripting Edition.

iMacros can read data directly from any Windows database using the Scripting Interface and a few lines of code.

This example code in Visual Basic Script connects to an Microsoft Access database:

```
' open database
set rs = CreateObject("ADODB.Connection")
rs.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" _
& mypath & "IIM-TEST-SUBMIT.MDB")

' use SQL to select information
```

```

sql = "select * from table1"
set rs = rs.Execute(sql)

' start iMacros
set iiml= CreateObject ("InternetMacros.iim")
iret = iiml.iimInit
iret = iiml.iimDisplay("Submitting Data from MS ACCESS")

' loop through result dataset
do until rs.eof
  'Set the variable
  iret = iiml.iimSet("-var_FNAME", rs.fields(0))
  iret = iiml.iimSet("-var_LNAME", rs.fields(1))
  iret = iiml.iimSet("-var_ADDRESS", rs.fields(2))
  iret = iiml.iimSet("-var_CITY", rs.fields(3))
  iret = iiml.iimSet("-var_ZIP", rs.fields(4))
  iret = iiml.iimSet("-var_STATE-ID", rs.fields(5))
  iret = iiml.iimSet("-var_COUNTRY-ID", rs.fields(6))
  iret = iiml.iimSet("-var_EMAIL", rs.fields(7))
  'Run the macro
  'Note: This is the SAME macro, as in the FILE-2-WEB-METHOD2.VBS example
  script!!!
  iret = iiml.iimPlay("wsh-submit-2-web")
  If iret < 0 Then
    MsgBox iiml.iimGetLastError()
  End If
  rs.movenext
loop



iret = iiml.iimDisplay("Done!")
iret = iiml.iimExit
WScript.Quit(0)

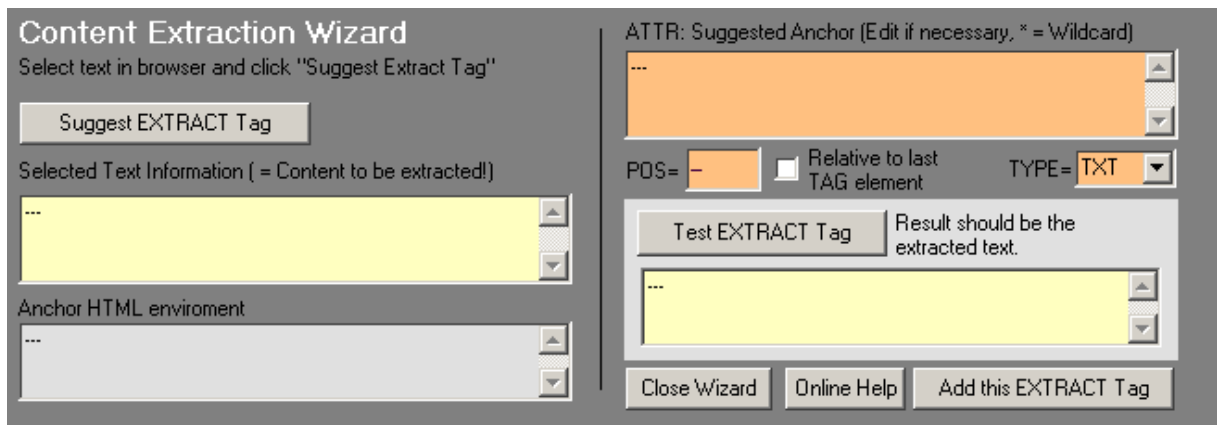
```

3.7 Extract Data from Websites

3.7.1 Extract single elements

Related example macro: *Demo-Extract*, *Demo-ExtractRelative*

iMacros can extract data from Web sites [iMacros Browser only]. Click on the  button while in recording mode to bring up the extraction wizard that will help you create the correct [EXTRACT](#)  command:



Note: Internet Explorer 6.0 or better must be installed in order to use the [EXTRACT](#)^[91] command.

The [EXTRACT](#)^[91] command and thus the extraction is controlled by three different parameters, the extraction anchor, the position and the type of extraction. The most important parameter is the extraction anchor. It contains information on the HTML code around the information which is to be extracted. You *must* use * at the end of the extraction anchor. If the HTML code given in the anchor appears more than once on a page, the position parameter determines which of the occurrences is extracted. The type of extraction determines if the result is plain text, HTML source code, an URL, an element's title or the alternative text of an image.

All extraction results can be accessed inside the macro through the built-in variable [!EXTRACT](#)^[122]. If this variable contains #EANF# (Extraction Anchor Not Found), the extraction was unsuccessful.

Results of multiple extractions in the same macro are separated by a [EXTRACT] tag in the [!EXTRACT](#)^[122] variable.

During manual replay of macros including [EXTRACT](#)^[91] commands in the iMacros Browser, the extraction result is displayed in a dialog window by default. This behaviour can be controlled by setting the built-in [!EXTRACT_TEST_POPUP](#)^[123] variable.

Some Background on HTML and Extraction

HTML is the language in which web sites are coded. The language consists of so-called tags, which determine how elements are formatted, displayed and aligned. Each HTML tag consists of two parts, an opening part and a closing part. All text between the opening and closing tags is affected by the directives the HTML tag implies. E.g. the following HTML snippet

This text is bold

will result in

This text is **bold**

i.e. the B tag is used to format text in bold face.

When extracting text with iMacros, the following procedure is applied:

- iMacros searches the HTML source of the currently active webpage for an occurrence of the extraction anchor
- If the anchor is found, all text between the opening HTML tag of the anchor and its equivalent closing tag is extracted
- If the anchor is not found, the result is #EANF#

Create Extraction Command

To define an `EXTRACT` command, proceed as follows:

Open the Extraction Wizard (`Extract Data` button on the Rec tab of the `control panel`^[97]).

Note: If the information you want to extract is inside a framed web site, you need to click inside the frame that contains the information you want to extract *before* opening the Extraction Wizard. This generates the `FRAME`^[94] command and marks the frame as active for the extraction.

In the browser window or frame, select the text that you want to extract.

Click the `Suggest EXTRACT Tag` button. The marked information will be displayed in the yellowish text area on the left. iMacros also creates a suggestion for the extraction anchor which is displayed in the orange text field on the right.

Click `Test EXTRACT Tag` to test run the extraction tag. The result of the generated extraction anchor will then be displayed in the yellow text area on the right side of the wizard. If the result is `#EANF#` (Extraction Anchor Not Found) you have to alter the extraction anchor in order to successfully extract data.

If you are satisfied with the result, click `Add this EXTRACT Tag` to add the `EXTRACT`^[91] statement to the macro.

Save Extraction Result

There are two methods to retrieve extracted data.

SAVEAS

You can save extracted data directly to a file by adding a `SAVEAS`^[106] `TYPE=EXTRACT` command manually to the macro. All items that were extracted before the `SAVEAS` command are saved to the specified file in one row like

```
"item1", "item2", "item 3", ...
```

As you can see, the `[EXTRACT]` tags are substituted by commas. The `SAVEAS`^[106] command erases the content of the `!EXTRACT`^[122] variable afterwards. With the next start of the macro or the next round of a loop, a new line is added to the file.

```
iimGetLastExtract()
```

You can also use the `iimGetLastExtract()`^[141] of the `Scripting Interface`^[38] to access the extracted data in your application. Potential `[EXTRACT]` tags are included in the returned string and can be used to separate different extraction results.

Unsuccessful Extraction

As said above, if the extraction was unsuccessful, i.e. the extraction anchor could not be found on the page, the `!EXTRACT`^[122] variable holds the string `#EANF#` (Extraction Anchor Not Found). However, the return value that informs you whether the execution of a macro was successful, is still positive (usually 2). The reason for this behaviour is that a macro can have many `EXTRACT`^[91] commands and often only one or a few of them do not find the extraction anchor. If you want to check if a particular `EXTRACT`^[91] command was successful, you just need to check if `#EANF#` is present in the returned string. Often, this can be very useful, for example if you use `EXTRACT`^[91] to check if a `keyword is present on a page`^[80]. A returned string containing `#EANF#` indicates that the keyword is not found.

Extraction of Dialog Text

To get the text of a dialog, use

```
SET !EXTRACTDIALOG YES
```

in the macro. Now, the content of a dialog is added to the extracted text, i.e. to the [!EXTRACT](#)^[122] variable.

Extracting From SELECT Elements

In HTML code, drop down lists are generated by a `SELECT` tag. For `SELECT` boxes, the currently active value is extracted. If you want to select *all* values of a drop down list, manually add `#ALL#` before the extraction anchor:

Select currently active values:

```
EXTRACT POS=1 TYPE=TEXT ATTR=<SELECT<SP>size=1<SP>name=main>*
```

Select all values in a list:

```
EXTRACT POS=1 TYPE=TEXT ATTR=#ALL#<SELECT<SP>size=1<SP>name=main>*
```


Extraction and the PRE Tag

Some web pages make use of a `<PRE> . . . </PRE>` tag in their HTML code. It marks the enclosed text as preformatted -- all the spaces and carriage returns are rendered exactly as you type them. The information enclosed in a `<PRE>` tag is extracted correctly (including the formatting!) by iMacros.

Thus if you transfer the extracted data via the [Scripting Interface](#)^[38] all formatting information is retained unchanged. The formatting is only changed on two occasions: Line breaks are removed when displaying the result in the test dialog box and when saving the result using the [SAVEAS](#)^[106] command. This is necessary to ensure proper formatting of the CSV formatted text file because in the CSV format, a line break would start a new line.

Trouble Shooting

Sometimes iMacros cannot suggest a proper extraction anchor automatically. In this case you can create one manually, enter it in the orange text area on the right side of the extraction wizard and test

it with the  button. Please read all the information in this Chapter to get a good overview over how the [EXTRACT](#)^[91] command can be tweaked manually.

3.7.1.1 The POS Parameter

This example is intended to shed some light on the use of the `POS` parameter of the [EXTRACT](#)^[91] command.

Consider the following HTML source code from which you would like to extract the **Text to be extracted** part:

```
<B>Hello World</B>
<B>Text to be extracted</B>
<B>Good Morning</B>
<B>Good Afternoon</B>
<B>(c) iOpus</B>
```

The extraction anchor is clearly `*` because the information is enclosed by a `B` tag. However, on this page this anchor can potentially match all 5 texts - we cannot further indicate which text to extract by tweaking the extraction anchor. But we can tell iMacros which occurrence of the extraction anchor to extract. Hence the correct `EXTRACT` command is:

```
EXTRACT POS=2 TYPE=TEXT ATTR=<B>*
```

As said before, you must use * at the end of the extraction anchor to tell iMacros that it should ignore the rest of the elements when searching for the anchor.

Starting with Version V4.30 you can also use [relative positioning](#)^[34] for the extraction anchor.

3.7.1.2 Extract with relative Positioning

Related example macro: *Demo-ExtractRelative*

When extracting data from a complex websites, the extraction can be made easier if you can tell iMacros to start the search for the extraction anchor after a specific point on the page (as opposed to start from the top, which is the default).

E.g., assume you want to extract data from a specific cell in a table, in this case the size of the land in the second table.

Result page 1

Transfers

| Deed Date | Book | Page | Deed Type | Deed Stamps |
|-----------|------|---------|------------|-------------|
| 5/9/2001 | 2034 | 845 | CorrDeed | |
| 3/23/2001 | 2021 | 460-461 | Trust Deed | |
| 6/3/1999 | 1882 | 463 | Warr. Deed | |
| 1/1/1901 | 0539 | 0321 | | |

Land

| Size | Units | Use / Dimensions |
|-----------|-------|------------------|
| 15000.000 | Sq.Ft | 100.000X150 |

Result page 2

Transfers

| Deed Date | Book | Page | Deed Type |
|-----------|------|------|------------|
| 4/4/2006 | 2677 | 501 | Warr. Deed |
| 9/15/2004 | 2454 | 337 | Warr. Deed |
| 6/18/1992 | 1443 | 978 | Warr. Deed |
| 2/20/1991 | 1373 | 270 | Warr. Deed |
| 1/1/1901 | 1196 | 0382 | |

Land

| Size | Units | Use / Dimensions |
|-----------|-------|------------------|
| 18200.000 | Sq.Ft | 130.000X140 |

Without relative positioning you would have to count the cell from the top of the page, including cell from other tables that come before the land table. Although the extraction wizard can do this for you, you run into problems as soon as the number of rows in a table are not constant as they are in the above example. The Transfer table of result 1 has four, that of result 2 has five row. Thus, an absolute position parameter like so

```
EXTRACT POS=8[33] TYPE=TXT ATTR=<TD>*
```

will potentially result in the extraction of an unwanted result.

With relative positioning you tell iMacros to search for the extraction anchor located *after* the position that is indicated by a [TAG](#)^[11b] command right before your [EXTRACT](#)^[9f] command. In our case we click on the table title "Land" before starting the extraction wizard to create a [TAG](#)^[11b] command. Note that this [TAG](#)^[11b] command does not click on any link, rather it only marks an element to indicate a position for the following [EXTRACT](#)^[9f] command. Relative positions are indicated with an R before the position number.

```
TAG POS=1 TYPE=B ATTR=TEXT:Land
EXTRACT POS=R1 TYPE=TXT ATTR=<TD>*
```

3.7.1.3 Using wildcards

The extraction anchor must end with the wildcard character *. However, you can also use the wildcard within the extraction anchor, as part of it. Consider the following HTML code and say, you want to extract the salary.

```
<li>
  <nobr>
    <font face="Verdana" size="-1">
      <b>Salary:</b>33,000.00 per year
    </font>
  </nobr>
</li>
```

One extraction anchor would for example be

```
EXTRACT POS=1 TYPE=TEXT ATTR=<FONT<SP>face=Verdana<SP>size=-1><B>Salary:</B>*
```

In this example, you can now substitute any parts of the anchor with the wildcard. The reason for doing this might be that the web site changes from time to time or you just need to have a more compact extraction anchor.

Suppose the font face changes every now and then, resulting in some unsuccessful extraction. You can easily change the anchor to

```
EXTRACT POS=1 TYPE=TEXT ATTR=<FONT<SP>face=*<SP>size=-1><B>Salary:</B>*
```

If the size also changes, you can change it to

```
EXTRACT POS=1 TYPE=TEXT ATTR=<FONT<SP>face=*<SP>size=*><B>Salary:</B>*
```

or even

```
EXTRACT POS=1 TYPE=TEXT ATTR=<FONT*><B>Salary:</B>*
```

If the extraction is successful, the extract text is "Salary: 33,000.00 per year".

3.7.1.4 Example: Keyword Anchor

Related example macro: *Demo-Extract*

Often you just want to extract information connected to a certain word on a web site. In this case, you can use a so-called keyword anchor, which is nothing else than the cunning use of [wildcards](#)^[35].

In this example, we would like to extract the information about the appearance of an object. As you can see from the example, this information is always connected with the word "Appeared".

FSBO 4-2-2, \$110 KCHAPEL CREEK 111-222-3333
 Fort Worth West Sale
 First **Appeared** in the FW Newspaper
 FSBO NICE AREA4-2.5-2 2243SF, \$139KMSID 222-555-9879
 Arlington Southwest Sale
 First **Appeared** in ABC Star
 SALE/LEASE condo, 3-2-2 enclosed patio comm. pool SW FW in
 Villas on the Bluff a gated comm. FSBO \$169.5k 555-1111-7997
 Condo/Townhome/Apt/DuplexSale
 First **Appeared** in the Star-Telegram

Using the Extraction Wizard, it would suggest a very general extract command, e.g.:

```
EXTRACT POS=29 TYPE=TEXT ATTR=<TD>*
```

While this might work, it is sensitive to small changes in the web page layout. If another table cell is inserted before this one, you will extract the wrong data. In this case, however, you can easily fine tune your extraction to make it more robust against web page changes using the keyword "Appeared" and wildcards:

```
EXTRACT POS=1 TYPE=TEXT ATTR=<TD>*Appeared*
```

Remember to reset the POS attribute since, with the "Appeared" part, it is the first occurrence.

3.7.1.5 Example: Data separated with

Related example macro: *Demo-Extract*

Sometimes you have line breaks in extracted information inflicted by the HTML tag
 like so:

```
<table>
  <tr>
    <td>
      John Smith<br>Main Street<br>Arlington
    </td>
  </tr>
</table>
```

On the web page, this might look like:

John Smith
 Main Street
 Arlington

You can only create one extraction tag for this line, i.e.

```
EXTRACT POS=1 TYPE=TEXT ATTR=<TD>*
```

So you can not create a separate extraction commands for name, street and city because these informations are not enclosed by separate opening and closing HTML tags. With the default TYPE=TEXT extraction, all parameters would be extracted into one line and are difficult to separate. The result would be

John Smith Main Street Arlington

To work around this problem, use the TYPE=HTM extraction. It preserves all HTML tags inside the text

so that the extraction result is:

```
John Smith<br>Main Street<br>Arlington
```

This result can be further processed and split with any programming or scripting language using the [Scripting Interface](#)^[53]. For example, in [Visual Basic Script](#) you can use the `Split` function:

```
MyArray = Split(extracted_string, "<br>")
```

MyArray will now have three elements, John Smith, Main Street and Arlington.

3.7.2 Extract complete tables

Related example macro: *Demo-Extract-Table*

To extract a complete table with only one command you can use

```
EXTRACT[91] POS=3 TYPE=TEXT ATTR=<TABLE*
```

The [!EXTRACT](#)^[122] variable now contains the entire information of the table. And iMacros has done more! It has put a `#NEXT#` tag between adjacent table elements and a `#NEWLINE#` tag at the end of every table row. These tags are automatically translated into commas and newlines when you use the [SAVEAS](#)^[106] `TYPE=EXTRACT` command, such that the following table

| Order# | Item | Price (US\$) |
|---------|--------|--------------|
| 331-445 | Book | 29.95 |
| 444-555 | CD-ROM | 15.00 |

will, when using this command

```
SAVEAS TYPE=EXTRACT FOLDER=* FILE=*
```

be saved in a CSV conform file looking like this

```
Order# , Item , Price (US$)
331-445 , Book , 29.95
444-555 , CD-ROM , 15.00
```

By default a comma (",") is used as separator. This can be modified by changing the entry for "CSVcomma=" in the iMacros settings file (File name: "iim.ini").

If you access the extracted information via the [Scripting Interface](#)^[38], you can easily use the separation tags to split the complete dataset.

3.7.3 Extract complete website

Since in the HTML language all web sites are enclosed by the `HTML` tag, you can extract a complete website including all HTML tags using

```
EXTRACT POS=1 TYPE=HTML ATTR=<HTML*
```

This can be very useful if you need to do your own parsing of the web site's contents.

To return only the body (main content) of a web page you can use

```
EXTRACT POS=1 TYPE=HTM ATTR=<BODY*
```

3.7.4 Save extracted data

Related example macros: *Demo-Extract*, *Demo-Extract-Table*

There are two methods to retrieve extracted data.

SAVEAS [PRO and SCRIPTING Edition]

You can save extracted data directly to a file by adding a [SAVEAS](#)^[106] TYPE=EXTRACT command manually to the macro. All items that were extracted before the SAVEAS command are saved to the specified file in one row like

```
"item1", "item2", "item 3", ...
```

As you can see, the [EXTRACT] tags, which are inserted to distinguish results from different [EXTRACT](#)^[91] commands, are substituted by commas. The SAVEAS command erases the content of the !EXTRACT variable afterwards. With the next start of the macro or the next round of a loop, a new line is added to the file.

iimGetLastExtract() [SCRIPTING Edition]

You can also use the [iimGetLastExtract\(\)](#)^[141] of the [Scripting Interface](#)^[38] to access the extracted data in your application. Potential [EXTRACT] tags are included in the returned string and can be used to separate different extraction results - see the included extract-2-database.vbs.

3.7.5 Extract & Scripting Interface

Related example scripts: *Extract-and-fill.vbs*, *Extract-2-file.vbs*, *Get-Exchange-Rate.vbs*

All extracted data can be sent to your code via the Scripting Interface. This gives you all the power of any programming language you choose to process the extracted information further or simply save it to a file.

Use the [iimGetLastExtract](#)^[141] command to return the extracted text if you used any [EXTRACT](#)^[91] commands within the macro.

The extracted text is returned as a string. Extracted information resulting from different [EXTRACT](#)^[91] commands are separated by [EXTRACT], e.g.

```
Text to be extracted[EXTRACT]Salary: 33,000.00 per year[EXTRACT]...
```

Remember: Using the [SAVEAS](#)^[106] TYPE=EXTRACT command will reset the contents of the [!EXTRACT](#)^[122] variable. Thus, using this command in a macro whose extraction result you wish to obtain via the Scripting Interface will result in an empty string in your application!

If you extract a complete table, the data from different columns is separated by #NEXT# and each table row ends with #NEWLINE#. You can easily use the separation tags to split the complete dataset. In Visual Basic Script, this would for example look something like

```
s = Replace(s, "#NEWLINE#", "" + vbCrLf + "")
```

```
s = Replace(s, "#NEXT#", """" + "," + """)
```

Example 1 - Split the returned string

The returned string is split to separate the results from different [EXTRACT](#)^[91] commands.

```
Dim data as String
Dim s as String
Dim ExchangeRate
iplay = iiml.iimPlay("wsh-extract")
If iplay = 1 Then
    data = iiml.iimGetLastExtract()
    ExchangeRate= Split(data, "[EXTRACT]")
    s = "One US$ costs " + ExchangeRate(0) + " EURO or " + ExchangeRate(1) +
    " British Pounds (GBP)"
    MsgBox s
End If
```

Example 2 - Keyword search

We want to find out if the word *iopus* exists on a web page. If yes, print the page. To make this example work, create the following macro and save it under the filename *mysearch.iim* in your Macros directory:

```
VERSION BUILD=3301125
'The keyword *is* the data extraction anchor!
EXTRACT POS=1 TYPE=TEXT ATTR=*iopus*
```

To print the web page, create the following macro and save it under the filename *print_this.iim* in your Macros directory:

```
VERSION BUILD=3301125
PRINT
```

Use the following Windows Script to control the macros:

```
set iiml= CreateObject ("InternetMacros.iim")
iret = iiml.iimInit()
iplay = iiml.iimPlay("mysearch")
extracted_text = iiml.iimGetLastExtract()
'test if keyword appeared on website.
If iplay = 1 Then
    if instr(extracted_text, "#EANF#") > 0 then
        MsgBox ("Sorry, keyword not found")
    else
        iplay = iiml.iimPlay("print_this")
    End If
End if
If iplay < 0 Then
    MsgBox "Error!"
End If
```

Note: You can also write directly to any Windows database. Please see the *extract-2-database.vbs* script for some example code. The script writes all results directly to a Microsoft Access database.

More Examples:

iMacros comes with several example scripts that demonstrate the `EXTRACT` command:

- extract-2-file.vbs
- extract-and-fill.vbs
- get-exchange-rate.vbs

The scripts are found in the `Examples\Windows Scripting Host` directory of your iMacros installation. More example scripts and test pages are available at <http://www.iOpus.com/iim/demo>

3.7.6 Extract Tech Tip

Here are some problems and workarounds for frequently asked questions:

Q: `EXTRACT` works while I am testing in the Extraction Wizard, but when I run the macro, Extract only returns #EANF# (Extraction Anchor not found).

A: Some websites are created dynamically from databases and the exact content of the website changes every time you visit a page. The solution is to replace the changing part of a link or extraction with the wildcard symbol.

Example:

Assume you searched for a product on a retailers site and the resulting page is a table of products, each with its own description and price tag, which are enclosed by the `A` HTML tag, like so:

```
<TR>
  <TD>
    <A class=price
href="/homes/homesforsale/view_details.jsp?advertID=14470882&listID=249
2&index=1&">
      Product 1, Price 1
    </A>
  </TD>
</TR>
<TR>
  <TD>
    <A class=price
href="/homes/homesforsale/view_details.jsp?advertID=14470882&listID=249
2&index=1&">
      Product 2, Price 2
    </A>
  </TD>
</TR>
<TR>
  <TD>
    <A class=price
href="/homes/homesforsale/view_details.jsp?advertID=14470882&listID=249
```

```
2&amp;index=1&amp">
    Product 3, Price 3
  </A>
</TD>
</TR>
[...]
```

Here is an extraction anchor as suggested by the Wizard for the first product:

```
EXTRACT POS=1 TYPE=TXT
ATTR=<A<SP>class=price<SP>href="/homes/homesforsale/view_details.jsp?advert
ID=14470882&amp;listID=2492&amp;index=1&amp">*
```

This command worked fine in the Wizard, but failed during the macro execution. Why? Because the `listID` part of the URL changes every time you visit the page. You can find this out, by running the Wizard twice (after refreshing the page in between) and comparing the extraction anchors. We also note that the variable `advertID` is probably the most important part of the link, since it defines the ad.

Solution:

Replace the changing `listID` number with `*`:

```
EXTRACT POS=1 TYPE=TXT
ATTR=<A<SP>class=price<SP>href="/homes/homesforsale/view_details.jsp?advert
ID=14470882&amp;listID=*&amp;index=1&amp">*
```

Actually, while you are at it, you can remove most static parts of the anchor as well. The result looks like:

```
EXTRACT POS=1 TYPE=HREF
ATTR=<A<SP>class=price<SP>href="*advertID=14470882*">*
```

If you want to cycle through all the ads on the page, you can do this as follows:

1. Replace the `advertID` number by an asterisk. Now, it will always find the matching extraction anchor.
2. To tell iMacros go for the second (third,...) product, change the `POS` parameter with a variable:

```
EXTRACT POS={{!LOOP}} TYPE=HREF
ATTR=<A<SP>class=price<SP>href="*advertID=*">*
```

During runtime, `{{!LOOP}}` takes on the values 1, 2, 3,... iMacros extracts the price on this page consecutively.

3.7.7 Asian Language Support

iMacros runs on all language version of Windows, including the so-called "double-byte" languages like Chinese, Japanese or Korean.

Data Extraction Tip:

Western (ANSI) characters can be extracted on any language version of Windows. In order to extract Asian characters correctly, please run iMacros on a Windows system that supports the language. Example: To extract Chinese characters, please run iMacros on the Chinese language version of Windows:




Example screenshot of iMacros data extraction on a Chinese Windows version.

3.8 Winclick

Related example macros: *Demo-Winclick*, *Demo-FileUpload*, *Demo-Flash*, *Demo-JavaScriptMenu*

The [WINCLICK](#)^[115] command is the ultimate solution if everything else fails! It simulates standard native mouse clicks within the browser window. This feature works with all web pages but you can only use it in the iMacros browser. It is activated during recording by clicking on the Click Mode: **Auto** button and selecting Windows Click from the upcoming dialog:

 Windows Click (Only iMacros Browser, Recommended to control Javascript menus, Java applets and Flash. Press CTRL+M to record mouseover event.

Typically, [WINCLICK](#)^[115] is used to automate web pages that contain non-HTML elements such as Java applets or Macromedia Flash elements.

What is the difference between the WINCLICK and the CLICK command?

- The WINCLICK commands operates on the visible part of the web page, just a like human would with a mouse. Thus WINCLICK X=1 Y= 400 will always click on the specific part of *the browser window*. If the web page is scrolled, the click hits another part of the web page.
- The CLICK command operates on the complete browser web page (HTML) only. So CLICK X=1 Y= 980 will always click *on the specific HTML element at this position of the page* regardless of whether the element is currently visible in the browser window or not. Whether or not the web page is scrolled, a specific X/Y combination always hits the same part of the web page.

You can use the HTML-based [TAG](#)^[110] or the [CLICK](#)^[89] commands to scroll a specific part of the web page into view and then use [WINCLICK](#)^[115] to operate using the now visible object (for example a Java applet).

The [WINCLICK](#)^[115] command can send keystrokes to the web browser via the CONTENT parameter. In addition to regular text, it can send special keys:

| | |
|-----------|-------------|
| ENTER | {ENTER} |
| TAB | {TAB} |
| DEL | {DEL} |
| BACKSPACE | {BACKSPACE} |
| LEFT | {LEFT} |
| RIGHT | {RIGHT} |

3.8.1 Trigger Mouse Over Events

Related example macros: *Demo-JavaScriptMenu*

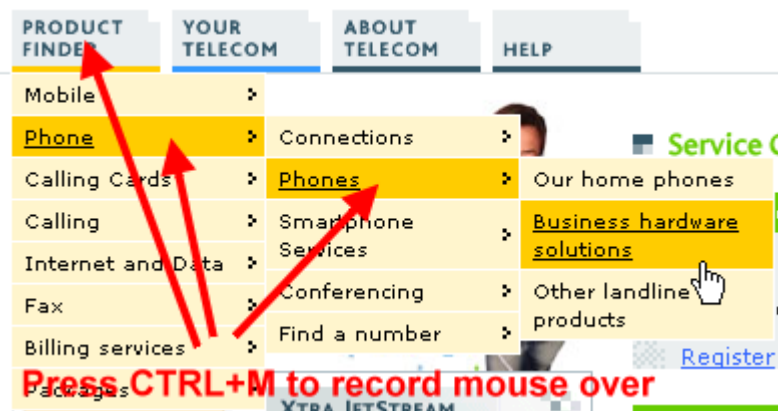
On some web pages menus are used that require the mouse pointer to be hovering over the menu in order to not fold it back. This can be achieved by instructing the [WINCLICK](#)^[115] command to send so-called mouseover events. You can instruct iMacros to do so via

```
WINCLICK X=200 Y=263 CONTENT=EVENT:MOUSEOVER
```

For example, you can use the mouseover event to automate Javascript-based menus. The same technique applies to mouseover and mousemove events in Java or Flash applets.

To record a mouseover event:

1. Select WINCLICK as Clickmode and make sure the box "Record Winclick at original speed" is **unchecked**.
2. Move the mouse over the menu(s) item you like to trigger with mouse over.
3. Press **CTRL+M** while keeping the mouse over the menu item.
4. The mouse over event is recorded.
5. Move the mouse to the next level and repeat .
6. Once you are at the final menu item, click to record a normal WINCLICK command.



Example: To automate selecting the "Business hardware solutions" item in the screenshot above, the following macro was recorded:

```
'Product finder (Level 1)
WINCLICK X=50 Y=135 CONTENT=EVENT:MOUSEOVER
'Phone (Level 2)
WINCLICK X=58 Y=157 CONTENT=EVENT:MOUSEOVER
'Phones (Level 3)
WINCLICK X=191 Y=162 CONTENT=EVENT:MOUSEOVER
'We reached the menu item to click (Level 4)
WINCLICK X=336 Y=209 CONTENT=
```

Note: Some Javascript menu systems have built-in timeouts, so it is important that the mouseover commands are sent with the right timing, neither too slow nor too fast. For most menus [SET !REPLAYSPEED](#)^[129] MEDIUM seems to work very well.

3.9 Response Time Measurements

Related example macro: *Demo-Stopwatch*

Related example script: *get-response-times.vbs*

The [STOPWATCH](#)^[109] command in iMacros allows you to measure the time that elapses between the first occurrence of the command in a macro (= stopwatch on) and the second occurrence (= stopwatch off). By using different identifiers in the [STOPWATCH](#)^[109] command you can create up to a 100 independent measurement points in your macro.

In order to do web site response measurements, you need to [insert](#)^[12] the [STOPWATCH](#)^[109] statements manually after you recorded your macro. For accurate measurements it is important to set the browsers [replay speed](#)^[129] to FAST so no artificial delays are added.

iMacros response time measurements always reflects the true user experience as they are measured using a real browser. Therefore response times measured by iMacros include loading times for browser plug-ins such as the Macromedia Flash Player or the Java runtime.

By default, the measured times are saved to the Downloads\ directory of your iMacros installation (e.g. C:\Program Files\iMacros\Downloads\). The default file name is macroName_stopwatch.csv. You can instruct iMacro to save the data to a custom file name by setting the built-in variable [!FILESTOPWATCH](#)^[124].

The values are comma separated (CSV format) so they can be viewed with any text editor, imported directly in Microsoft Excel or viewed by any other software you use. Additional information about the date and time of the measurements and the calling macro will be added to the response times.

In this example we measure response times of different parts of the iOpus homepage:

```
VERSION BUILD=4230323
SET !FILESTOPWATCH mydata.csv
STOPWATCH ID=total
URL GOTO=http://www.iopus.com/iim/
STOPWATCH ID=1
TAG POS=1 TYPE=A ATTR=HREF:http://www.iopus.com/iim/compare
STOPWATCH ID=1
STOPWATCH ID=store
TAG POS=1 TYPE=A ATTR=TEXT:US$<SP>149
TAG POS=1 TYPE=INPUT:SUBMIT FORM=NAME:order
ATTR=NAME:ORDER_PRODUCT_NOW&&VALUE:Order<SP>Now
TAG POS=1 TYPE=A ATTR=HREF:http://www.iopus.com/store
STOPWATCH ID=store
STOPWATCH ID=total
```

This macro will create the following data in the mydata.csv file - obviously the response times will be different when you replay this macro. The format of the file is:
 YYYY/MM/DD, HH:MM:SS, Macro name, ID, time (s)

```
"2004/08/3", "11:56:23", "mymacro", "1", "1.272"
"2004/08/3", "11:56:32", "mymacro", "store", "8.943"
"2004/08/3", "11:56:32", "mymacro", "total", "10.21"
```

For more information, please see the [Tips for Accurate Web Response Time Measurements](#)^[45].

3.9.1 Automating Response Time Measurements

Related example script: *get-response-times.vbs*

If you want to automate response time measurements, it is likely that you will call iMacro from another application. Instead of writing the [response time](#)^[44] to a log file, the data can be sent to your application via the [Scripting Interface](#)^[53]. Simply use the [extract](#)^[38] feature. The last recorded response time value is stored in the built-in [!STOPWATCHTIME](#)^[130] variable. You can use this variable as follows to transfer the response time data to the calling script or program:

```
VERSION BUILD=4230323
SET !FILESTOPWATCH NO
STOPWATCH ID=total
URL GOTO=http://www.iopus.com/iim/iim/demo/v4
STOPWATCH ID=1
TAG POS=1 TYPE=A ATTR=HREF:http://www.iopus.com/iim/compare
STOPWATCH ID=1
SET !EXTRACTADD {{!STOPWATCHTIME}}
STOPWATCH ID=store
TAG POS=1 TYPE=A ATTR=TEXT:US$<SP>149
TAG POS=1 TYPE=INPUT:SUBMIT FORM=NAME:order
ATTR=NAME:ORDER_PRODUCT_NOW&&VALUE:Order<SP>Now
TAG POS=1 TYPE=A ATTR=HREF:http://www.iopus.com/store
STOPWATCH ID=store
SET !EXTRACTADD {{!STOPWATCHTIME}}
STOPWATCH ID=total
SET !EXTRACTADD {{!STOPWATCHTIME}}
```

Thus the data is added to the !EXTRACT variable. Its contents can be obtained via the [iimGetLastExtract](#)^[141] command. In the above example, it contains the three recorded response times (Example string):

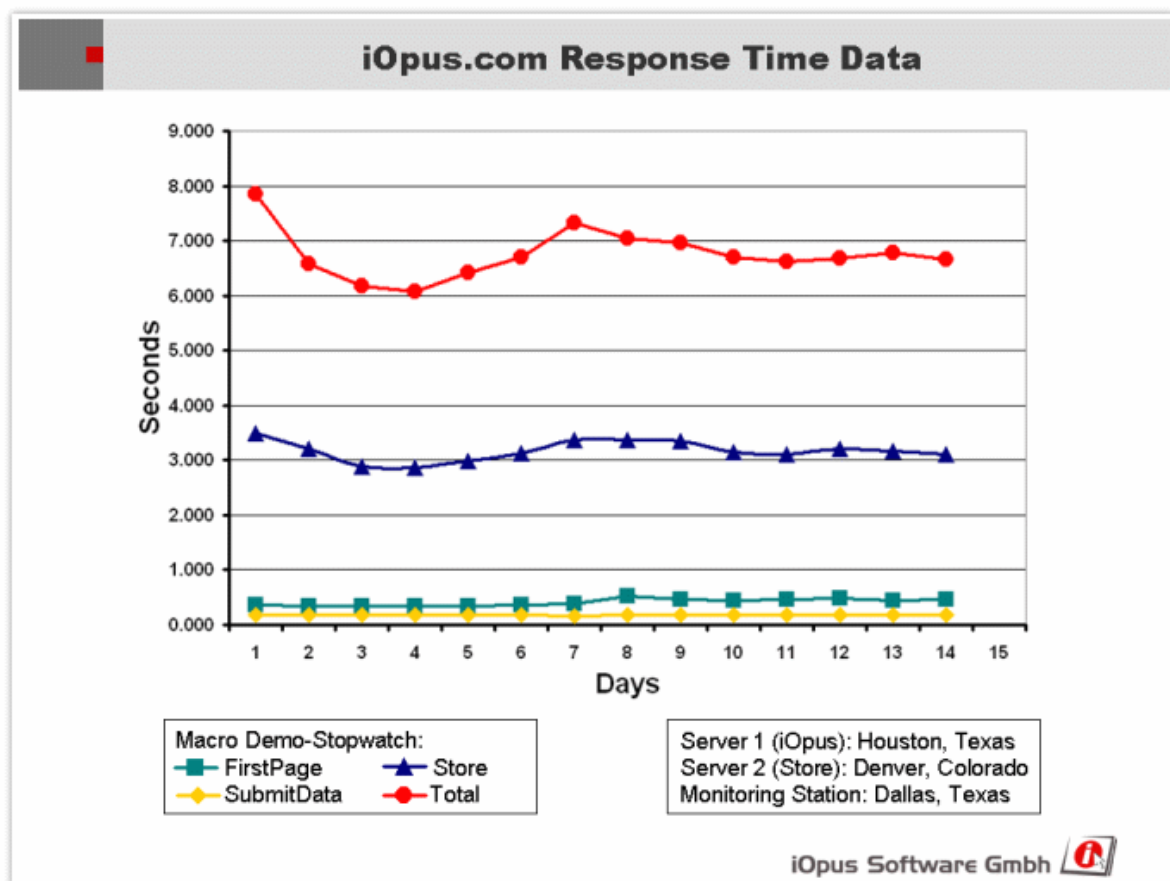
```
1.272[EXTRACT]8.943[EXTRACT]10.21[EXTRACT]
```

Note: SET !FILESTOPWATCH NO instructs iMacros *not* to create a response time log file. This is useful if you intend to only return the values via the Scripting Interface.

3.9.2 Tips for Accurate Web Response Time Measurements

- Add a [CLEAR](#)^[88] statement to your macro. This way you can make sure that the browser cache is cleared before each run. Otherwise, iMacros might read the web pages from the cache and not the web server, which would most likely result in lower response times. Whether the cache is actually used, depends on the Internet Explorer settings.

- **Run the measurements as a loop and average several runs.** The "internet speed" can fluctuate from minute to minute even on a fast connection. Therefore differences between each measurement run are normal. To get stable results, it is good practice to average several runs. Several common programs such as Microsoft Excel can create averages automatically for you.
- If you compare results between different PC's, please keep in mind that the accuracy depends on the accuracy of the PC clock. This applies to all software that does time measurements on a PC.
- Under normal conditions the processor speed does not influence the measured response times. Only if the PC is so slow that the web page rendering of the browser is slowed down, will the CPU speed have an influence on the measured response time. iMacros response time measurements **always** reflect the **true user experience** as they are measured using a real browser and the original browser plug-ins such as Macromedia Flash Player or SUN Java runtime.
- Sample measurements with a modified Demo-Stopwatch macro running on the iOpus.com Dallas monitoring server:



3.10 Change User Agent

Related example: *Set-User-Agent.vbs* (VBS script)

Every time you access a web site, the browser you use sends a string to the web server containing information about your operating system and the browser you are using. This string might, for example, look like this:

Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)

Sometimes it is desirable to pretend to be a different user agent because some web sites change in behaviour or appearance depending on the user agent. iMacros can simulate all user agent strings with the `-useragent` command line switch. The command line switch can also be used in [`iimInit`](#)^[142] command of the the Scripting Interface:

```
iret = iim1.iimInit ("-useragent " "Nokia6230/2.0+(04.43) " " ")
```

If your user agent contains spaces, please use double quotes ("") around it.

You can see the current user agent of iMacros at <http://www.iopus.com/imacros/demo/v5/user-agent.htm>

3.11 Filter

Filtering is a new feature that allows you to change data on the website before it reaches the browser. Currently only the `TYPE=IMAGES` filter is supported. To start the filter and remove images, add this line to your macro:

```
FILTER TYPE=IMAGES STATUS=ON
```

To stop the filter:

```
FILTER TYPE=IMAGES STATUS=OFF
```

If enabled, it removes all references to images from the html source and thus speeds up page loading. *The filter command works for all macros until the filter is switched off or the iMacros Browser is closed.*

Currently the support for filtering is experimental. If you need any other data filtered, please [let us know](#) what kind of filter you would like to see added.

3.12 Send Email

There are three ways to send emails with iMacros:

1. iMacros can fill out an online form that sends the email, for example a form on your website similar to the "Email this story" link on Yahoo:

<http://mtf.news.yahoo.com/mailto?url=http%3A//biz.yahoo.com/>

2. You can use a local [command line SMTP mail sender](#). Using for example a batch file that executes the macro, you could call the command line mail program depending on the status of the macro execution.

3. For information on how to send email with the Windows Scripting Host (WSH) take a look at the following examples:

<http://www.rgagnon.com/wshdetails/wsh-0018.html> or

<http://www.winscripiter.com/wsh/internet/simpleemail.asp>

Note: iOpus is not affiliated with the listed websites, they are only provided as an example.

3.13 Error Handling

iMacros can handle *all* errors that occur during replay. Since replay of macros can be achieved by different means, the error management is different as well.

Macro level

You can define a macro to play after an error. This can be done globally through the Error Handling tab of the Options Dialog, or with [SET](#)^[107] [!ERRORMACRO](#)^[122] yourmacro within a macro, which will override any global setting.

You could also decide to ignore errors with [SET](#)^[107] [!ERRORIGNORE](#)^[121] YES.

Related Example Macro: Demo-SetErrorMacro

Batch file^[50] level

The variable %ERRORLEVEL% is filled after the macro is completed. Values greater than zero indicate success, negative values indicate a problem.

Related Example Batch File: Example-Errorlevel.bat

Scripting^[53] level

The command [iimPlay](#)^[143] returns a status value. Values greater than zero indicate success, negative values indicate a problem.

The error codes returned on the batch and scripting level are the same codes that are displayed in the iMacros software itself. They allow a fine reaction on every possible problem iMacros can encounter. Scripting Interface specific errors are discussed [here](#)^[145].

Related Example Script: Combine-Macros.vbs

3.13.1 Error Codes

General classification:

-1xx : Installation and Setup related error codes

-2xx: Errors during Macro recording

-3xx: Errors during Macro replay

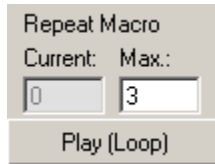
-4xx: Errors specific to the IE Plug-In

-5xx: Errors specific to the [Scripting Interface](#)^[145]

A detailed list of error codes is available online at <http://www.iopus.com/iim/support/error-codes.html>

4 Automation

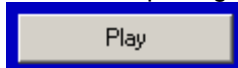
Several tasks you can automate using iMacros require multiple or regular execution, like filling in an online form with many datasets or regularly downloading a status report. iMacros has several features to support this kind of automation:



- [Button](#)^[27] or Shortcuts [All Editions]
- [Command line interface](#)^[50] [PRO and SCRIPTING Editions]
- [Scripting Interface](#)^[53] [SCRIPTING Edition]

4.1 Shortcuts

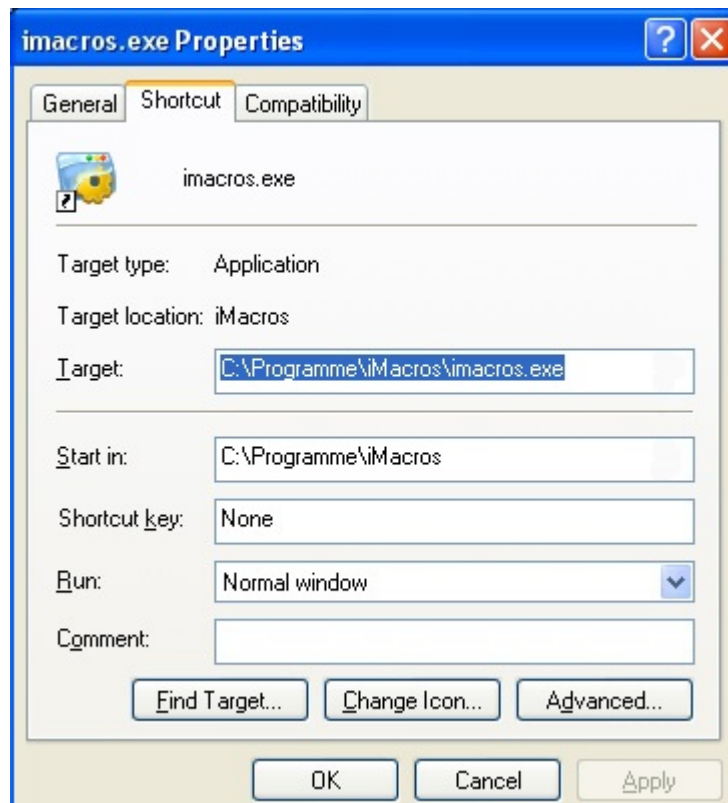
Instead of opening the iMacros Browser, marking the macro you want to play and clicking the



button, you can create a shortcut to start iMacros and automatically run a predefined macro.

To create a shortcut:

- Locate `imacros.exe`. This is the iMacros Browser. Typically, this file is located in `C:\Program Files\iMacros`.
- Right-click on `imacros.exe` and select the "Create a Shortcut" option.
- A new file called `Shortcut to imacros.lnk` will be created in the same folder. Select the file and right click on it.
- Select "Properties" and the following dialog will come up:



- In the box **Target** you find an entry similar to `C:\Program Files\iMacros\imacros.exe`.
- Change this to `"C:\Program Files\iMacros\imacros.exe" -macro yourmacro -noexit`. You must use quotation marks around the command.
- Note: Do **not** change the path in the box **"Start in"**.
- Ready! You can now move this shortcut to a convenient place (like the Desktop) and start iMacros by simply double-clicking on the shortcut.

For a detailed explanation of the various command line options please see [here](#)^[132].

4.2 Batch Files

Related example batch: *Examples\Batch Files*

The iMacros Browser (included in PRO and Scripting Edition) supports several [command line options](#)^[132] that control the behaviour of the browser. This browser can be called from batch files. Batch files are very simple programs that allow structures the iMacro macro language does not support directly. Such structures include conditional programming (`if...then...else...`), loops (`for...do...`) and file input/output. They consist of a series of commands, can be created and edited by any editor (e.g. Notepad) and have the file ending `.bat`. Batch files can be started the standard way, by double-clicking on the file, but also through the [Scheduler](#)^[52], which is a part of the Windows operating system by default.

Please note that file [shortcuts](#)^[49] can be looked at as nothing else than a batch file that consists of only one line, the command line calling the program.

The most important part of a batch file that starts iMacros is the command line which starts the browser. As mentioned before, this command supports several [command line options](#)^[132] that control

the behaviour of the browser. The general command line syntax is

```
imacros -command_line_switch1 switch_value1 -command_line_switch2
switch_value2 ...
```

A command line that starts the iMacros Browser, executes the macro called `YourMacroName` and passes a value to the built-in variable `!VAR1` look like this:

```
imacros -macro YourMacroName -var1 Hello[SP]World!
```

After completing the macro, the iMacros Browser automatically stops and closes. Note that in command lines you need to use the square brackets instead of the usual pointy brackets (`<` and `>`) to escape a whitespace (`[SP]`) or linebreak (`[BR]`). This is because `>` and `<` are used for file input/output operations on the Windows command line and thus the command does not execute correctly.

You can now use batch files to conveniently execute several iMacros in a sequence:

```
echo Start iMacros batch file
imacros -macro FormExampleMacro
imacros -macro Check_Altavista
imacros -macro Buy_Now!
echo Done!
```

In the above macro the batch command `echo` is used to print messages to the standard output.

Other important command line switches are those setting variables. You can set the built-in variables `!VAR1`, `!VAR2` and `!VAR3` using the following switch:

```
imacros -var1 iOpus[SP]iMacros
```

With the command line switch `-var_varname` you can create your own custom variable. Imagine, you want to call a macro (`searchEngine.iim`) that enters the content of the variable `SEARCHSTRING` into a search engine and presses submit. You can set the search value via the command line:

```
imacros -macro searchEngine -var_SEARCHSTRING iOpus[SP]iMacros
```

If you want iMacros to read a value from the command line use the [CMDLINE](#)⁹⁰ command in your macro.

```
CMDLINE !VAR1 http://www.iOpus.com/iim.htm
SET !VAR2 nobody@nospam.iOpus.com
TAG TYPE=INPUT:TEXT FORM=NAME:f1 ATTR=NAME:name CONTENT=Tom Tester!VAR1
```

Check on the return value at batch level

The iMacros executable `imacros.exe` sets the predefined batch file variable `ERRORLEVEL`. The value is either 1 if the macro was completed successfully or negative if an error was encountered. `ERRORLEVEL` is used almost exclusively with a conditional construct, e.g.:

```
imacros -macro searchEngine -var_SEARCHSTRING iOpus[SP]iMacros
IF NOT %ERRORLEVEL% == 1 ECHO Problem encountered
```

For more details please see the included example batch file "Example-Errorlevel.bat".

4.3 Schedule Tasks

iMacros is fully compatible with the built-in Windows Task Scheduler and any other task scheduler.

The Microsoft Task Scheduler is part of your Windows system. It "hides" in the "Control Panel" folder, which is located in the "My Computer" folder on your desktop.

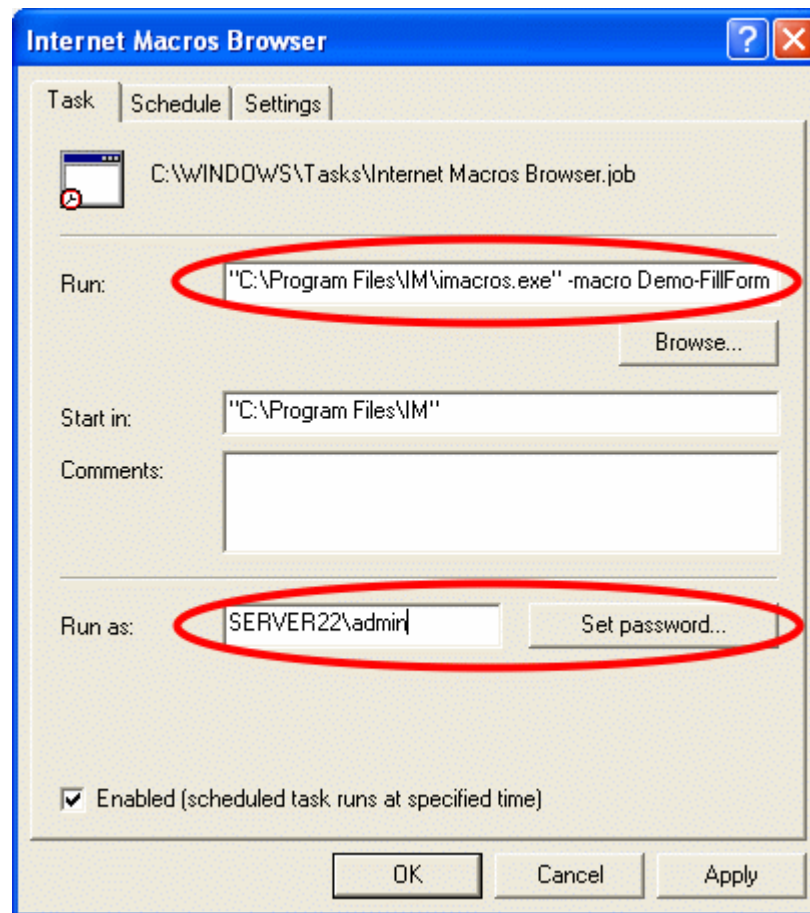
By using Task Scheduler, you can schedule tasks such as iOpus iMacros or system tools like the Disk Defragmenter to run at a defined time that is most convenient for you. The Task Scheduler starts each time you start Windows and runs in the background, checking if any scheduled task is due. With Task Scheduler, you can schedule a task to run daily, weekly, monthly, or at certain times such as system startup.



To use the scheduling service on **Windows 98, NT, 2000**, click on the My Computer icon located on the Desktop. Then double-click on Control Panel to get to the Scheduled Tasks folder.

In **Windows XP and Server 2003** you can access this from the Start Menu and clicking on Settings and then Control Panel to Scheduled Tasks.

Double-click Add Scheduled Task. Follow the instructions in the Add Scheduled Task Wizard. Select the Open advanced properties for this task for more set up options.



What you enter at the "Run" file could look like this:

```
"C:\Program Files\InternetMacros4\imacros.exe" -macro YourMacro -tray -loop 50
```

Important: The actual command must be *inside* quotation marks whereas the command line options must be placed *outside*.

Important: If you want your task to run even if no user is logged in, you must enter a user name and password in the "Run As" box.

Alternatively you can point to a [Batch file](#)^[50] or [Windows Scripting \(VBS\)](#)^[53] file that contains the specific instructions to start the software. Thus you can conveniently execute several commands in a sequence.

4.4 Control via Scripting Interface

If you own the Scripting Edition, iMacros automatically installs the [Scripting Interface](#)^[139]. Using these powerful commands, you can control iMacros with any Windows programming language that supports

using COM objects.

Almost all Windows programming languages support this technology, including the free [Windows Scripting Host](#), Visual Basic 6, Visual Basic .NET, C#, Java, Perl, Python, C++, ASP, PHP, ASP.NET. On the [iOpus homepage](#) many examples with different programming languages can be found.

This Chapter will provide some examples of how to use the Scripting Interface. The examples will be using [Visual Basic Script](#)^[54], [Visual Basic](#)^[55], [starting iMacros from a web site](#)^[58] and [starting iMacros as a Windows Service](#)^[57]. On our web site we have [tutorials](#) for many other programming languages.

4.4.1 Example Windows Scripting Host

Related example scripts: *Examples\Windows Scripting Host*

The Window Scripting Host interprets programs written in a language called Visual Basic Script, which is related to Visual Basic and the macro languages of the Microsoft Office package (VBA). Visual Basic Script files can be created and edited with any editor (e.g. Notepad), are executed by double-clicking them and have the file ending .vbs.

The iMacros Browser is controlled by the Visual Basic Script by calling commands of the [Scripting Interface](#)^[139].

The following example creates an instance of the iMacros Browser, sets some variables and plays a macro. The return value of the macro is then checked for errors. To run this example, copy this text in a file with the ending .vbs, e.g. test.vbs. After double-clicking the file, iMacros will start in tray mode since in line 4 the command line switch [-tray](#)^[136] is activated. This means that an iMacros icon will appear in the system tray during replay. You can maximize iMacros by double-clicking this icon.

```
'initialize Scripting Interface
Set iim1 = CreateObject ("InternetMacros.iim")
i = iim1.iimInit()

' setting variables
i = iim1.iimSet("-var1", "Tom Tester")
i = iim1.iimSet("-tray", "")

' displaying message
i = iim1.iimDisplay("This is a test")

' play macro
i = iim1.iimPlay("myfirstmacro")

' check success
If i > 0 Then
    s = "Everything OK"
Else
    s = iim1.iimGetLastError()
End If
MsgBox s

' exit iMacros
i = iim1.iimExit()
```

4.4.2 Scripting Example Visual Basic.NET

Related example scripts: *Examples\Visual Basic*

You can then use the function of the [Scripting Interface](#)^[139] to control iMacros. The following example is part of a Visual Basic project that creates an iMacrosBrowser instance, plays a macro and checks for errors - this example assumes the existence of a function called `void Log(String logString)`, which logs messages.

```
Imports Status = InternetMacros.Status

Public Class Form1
    Inherits System.Windows.Forms.Form

    Private m_app As InternetMacros.App

    Private Sub startIMacros()
        Const cmdTimeout = 60

        Dim s As InternetMacros.Status

        m_app = New InternetMacros.App

        s = m_app.iimInit("", True, "", "", 5)

        s = m_app.iimPlay("demo-extract", cmdTimeout)

        If s > 0 Then
            Log("Macro completed ok")
        ElseIf s < 0 And s > -100 Then
            Log("Interface problem: " + CStr(s))
        Else
            Log("Macro problem: " + CStr(s))
        End If

        s = m_app.iimExit()

    End Sub
End Class
```

4.4.2.1 Intellisense Support

Related example scripts: *Examples\Visual Basic*

For full Intellisense support for all [Scripting Interface commands](#)^[139] in your .NET projects (C#, VB.NET, ASP.NET and others) or in Visual Basic 6.0, you need to add the iMacros interface (iimInterface.dll) reference to your project.

1. Here is how to do this in **Visual Studio 2003/2005 (.NET)**

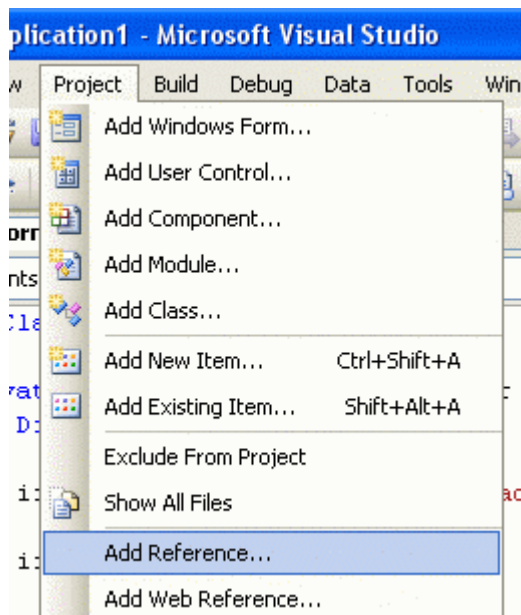


Fig 1: Select the iMacros Type Library component

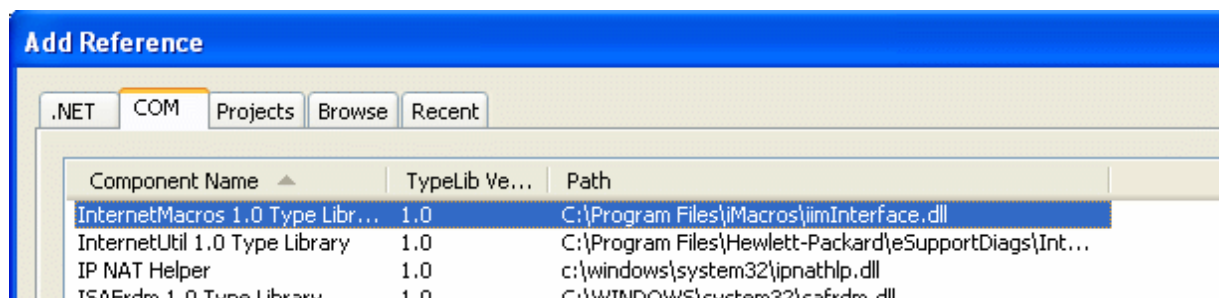


Fig 2: Select the iMacros Type Library component

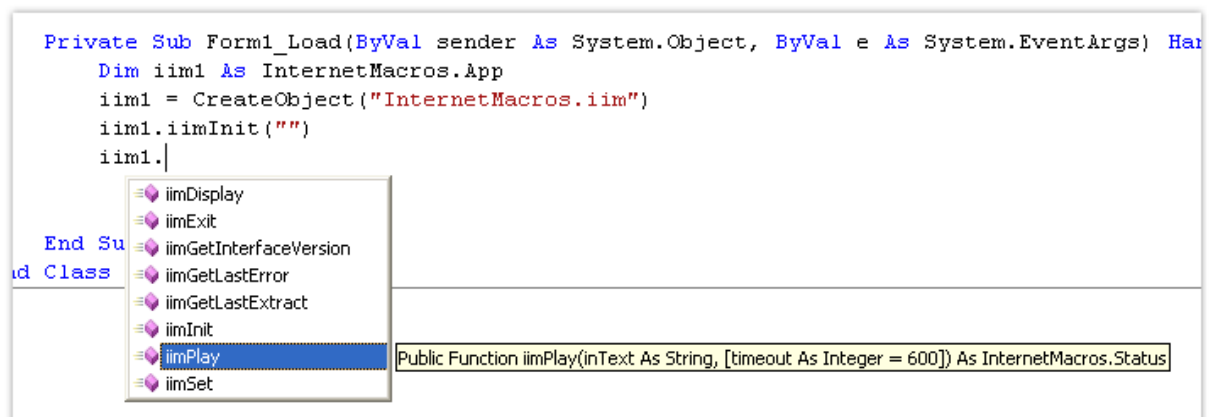


Fig 3: Display all Scripting Interface commands with Intellisense.

2. Here is how to do this for **Visual Basic 6**

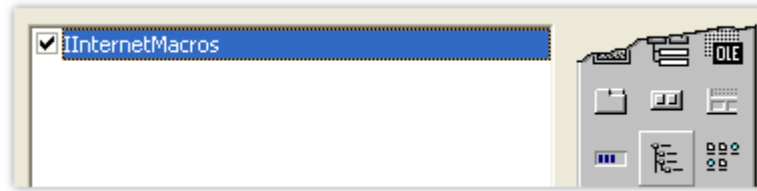


Fig 4: Select the iMacros component

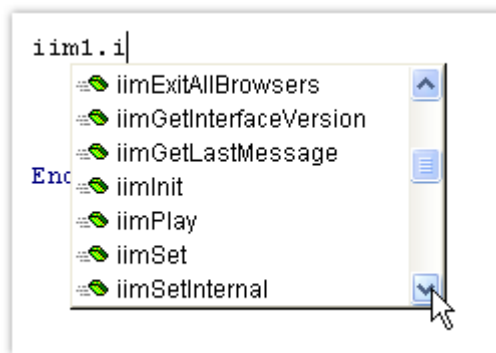


Fig 5: Display all Scripting Interface commands with Intellisense.

4.4.3 Run iMacros under a different user account

We created the "iimRunner.exe" method as a very easy and secure method from a very restricted user account (such as the ASP.NET user account or a Windows Service).

Start the "iimRunner.exe" module inside the account, that you want to use iMacros with. In the following, we assume that is a regular user account called "user1". You can use for example the [Windows task scheduler](#) for this purpose: Select the option to start iimRunner.exe as soon as the computer boots.

Inside your ASP script you only need to tell iimInit to use iimRunner to start iMacros.exe. This is done by adding the flag "-runner":

```
i= iim1.iimInit("-runner")
```

This is all that you need to do. Now the iMacros Browser can be controlled as before via iimPlay, iimExit etc.

Limit the number of running instances: As a special feature of iimRunner.exe, it can control the number of iMacros instances that are allowed to run in parallel. This is done by changing the MaxNumberOfInstances parameter in the simple.config file. If the max. number is reached, iimInit returns a -7 error code.

4.4.4 Start as Windows Service

Windows Services is a topic for experts only. If you do not know what a Windows Service is, there is a very good chance that you do not need this feature. If you are merely looking for a way to execute iMacros on a regular basis, the [Windows Scheduler](#)^[52] will do the trick for you.

In the following, we assume that you are familiar with the basic concept of a Windows Service.

You can start iMacros via any application that runs as Windows Service - more information on services can be found [here](#). The advantage of using a Windows Service is that the application can run even if no user is logged in. Due to a restriction enforced by Windows on services, a service program can either be interactive (i.e. have a Console, read keyboard input, etc) or have network access - but not both at the same time. Since iMacros needs the ability to use the network more than user input, you need to provide the user name and password of a normal Windows account.

If you use iMacros inside an application that runs as a Windows Service (as opposed to running under a regular user account) you need to provide a user name and a password:

There are two methods to do this:

(1) [Recommend]: Use "iimRunner.exe" to start iMacros under a regular user account. This method is very easy to use and avoids all complications typically associated with a Windows service

(2) Provide a password when you use [iimInit](#)^[142]:

```
int ret_code =  
    iimInit( String command_line, boolean start_browser, String run_as_user,  
            String run_as_password, String run_as_domain )
```

You can also provide the password encrypted. iMacros detects automatically if the password is in plain text or encrypted form. To encrypt your password, record a macro where you enter the password into a password text field on a web page (with [security settings](#)^[13] to STOREKEY or TMPKEY) and then copy and paste an encrypted password.

Running iMacros as a Windows service does not affect the concept of a user session, because iMacros - while started by a service - effectively runs under the user account which you use in [iimInit](#)^[142].

4.4.5 Start from Web Page

The following code example shows you how to start a macro from within a web page using [Visual Basic Script](#)^[54] and the [Scripting Interface](#)^[139]:

```

<html>
<head>
<SCRIPT ID=clientEventHandlersVBS LANGUAGE=vbscript>
<!--
Sub test()
  Dim WSHShell, iim1, iret
  Set WSHShell = CreateObject("WScript.Shell")

  MsgBox ("This example starts a macro from within a web page")

  Set iim1 = CreateObject ("InternetMacros.iim")

  iret = iim1.iimInit()
  iret = iim1.iimDisplay("Start Macro now")

  iret = iim1.iimPlay("Demo-Frame")
  If iret < 0 Then
    MsgBox iim1.iimGetLastError()
  End If

  iret = iim1.iimDisplay("Done!")
  iret = iim1.iimExit()
End Sub
-->
</SCRIPT>
</head>
<body>
<p><a onclick=test() href="#">Click here to start script</a></p>
</body>
</html>

```

To test this code, simply save it as .HTM file and view it in your web browser. Make sure to allow the execution of scripting content on web pages.

In this example, iMacros needs to be installed locally. It will run on the client. This is contract to the ASP example, where iMacros runs on the server (invisible to the user).

4.4.6 ASP/ASP.NET/PHP

In the following text, we will use ASP as an example. But the method described here works for ASP,ASP.NET or in general as a method for controlling iMacros from a very restricted user account.

When using ASP (etc.) iMacros will run on the **server**, not on the client PC.

Problem:

The problem of running iMacros from an ASP page is that by default all programs started by an ASP page have only the rights of the ASP user, which is a very restricted account. An ASP account is significantly more restricted than even the "Guest" user of a machine. However, iMacros needs at least the rights of a "Guest" account or "Limited account" in order to work correctly.

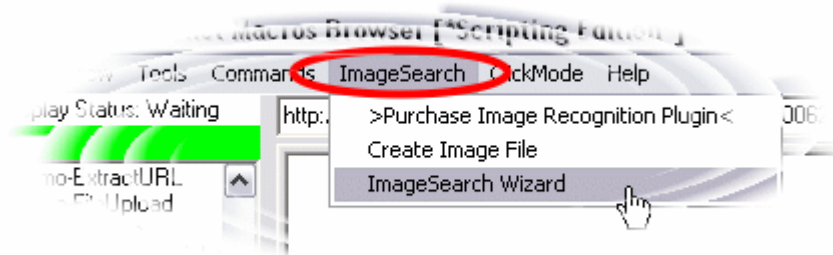
Solution:

Use the [iimRunner method](#)⁵⁷ and start the "iimRunner.exe" module inside the account, that you want to use iMacros with.

5 Image Recognition Plugin

The [Image Recognition Plugin](#) can be purchased separately and adds two new, extremely powerful commands to the PRO and Scripting Edition. This addition is highly recommended if you need to test web sites with non-html based functions, such as Macromedia Flash (Shockwave) or Java applets. Unlike [WINCLICK](#)^[42] the new [IMAGESEARCH](#)^[96] and [IMAGECLICK](#)^[95] commands do not rely on the coordinates of an element but on its visual appearance. Thus buttons and input boxes can be found even if they move around on the screen.

After the installation of the Image Recognition Plugin (IMImage.dll) a new menu entry appears in the iMacros browser menu:



The menu entries allow you to create and test input images for the new image recognition commands.

Note: In the unregistered trial version of the plugin, a "Purchase" menu item offers purchase and pricing information. This menu item is not displayed in the full version of the plugin.

5.1 Search and Click Images/Buttons

The commands [IMAGESEARCH](#)^[96] and [IMAGECLICK](#)^[95] are active after the Image Recognition Plugin installation.

Search for an image on a website

The command to search certain images on a website is called [IMAGESEARCH](#)^[96] and has two parameters, `IMAGE` and `CONFIDENCE`. The command works by searching for the appearance of an image on the web site, which is saved locally on your hard drive. The parameter `IMAGE` sets the image that the Plugin searches for. The parameter `CONFIDENCE` tells the Plugin how close the image must match the image on the website. A confidence level of 100 means that the input image and the image on the website must be 100% identical - an exact match; otherwise the [IMAGESEARCH](#)^[96] command generates an ERROR. A confidence level of less than 100, e.g. 95, allows for some deviation in the color of the image, or a change in the image background or foreground colors.

Click (or send text) to a certain area of the website (defined by an image)

The command to click or send content to an image on a web site is called [IMAGECLICK](#)^[95]. This command has three parameters, `IMAGE`, `CONFIDENCE` and `CONTENT`. The command works by searching for the appearance of an image on the web site, which is saved locally on your hard drive. If this image, supplied by the `IMAGE` parameter, is found, the `CONTENT` is sent to it. The parameter `CONFIDENCE` has the same function as in the [IMAGESEARCH](#)^[96] command.

The [IMAGECLICK](#)^[95] command is a combination of [IMAGESEARCH](#)^[96] and [WINCLICK](#)^[42]. The following example shows the connection:

```
IMAGESEARCH[96] IMAGE=pic1.bmp CONFIDENCE=95
WINCLICK[42] X={ { !IMAGEX[125] } } Y={ { !IMAGEY[125] } } CONTENT=
```

The internal variables [!IMAGEX](#)^[125] and [!IMAGEY](#)^[125] are the X/Y coordinates of the middle of the position where the "pic1.bmp" image was found. They are set by the [IMAGESEARCH](#)^[96] command.

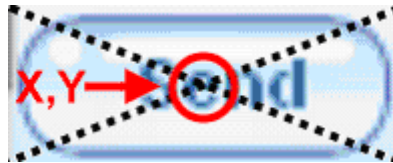


Fig 1: Coordinates returned by [IMAGESEARCH](#)^[96]

Some comments on the CONFIDENCE parameter:

A confidence level of 100 means that the input image must match the image found on the web page 100%. The images are identical on the pixel level. This also means that no deviations in size, color or shape are allowed. The search algorithm used for `CONFIDENCE=100` is *very fast*. It is recommended that use 100 if you do not expect any changes in the image. Typical search times are less than a second on a PC with a 2.5 GHz CPU.

A confidence level of less than 100 means that the input image does not have to match the image found on the web page exactly. Deviations in color or shape are allowed. The lower the confidence level, the more deviations are allowed. If you simply want to ignore color changes, a `CONFIDENCE` level of 95 is recommended. If you lower the confidence level too much, iMacros will detect the wrong image as an acceptable match. For typical web testing applications, levels between 50 and 99 have proven to be useful. Since, the search algorithm uses advanced image recognition technology it is not as fast as a `CONFIDENCE=100` search. Also, the search time increases with input image size. Typical search times are between 1-3 seconds on a PC with a 2.5 GHz CPU.

5.2 Create Images for IMAGESEARCH

To create a new image for use with the [IMAGESEARCH](#)^[96] or [IMAGECLICK](#)^[95] commands, you can use any screenshot tool and any image editor.

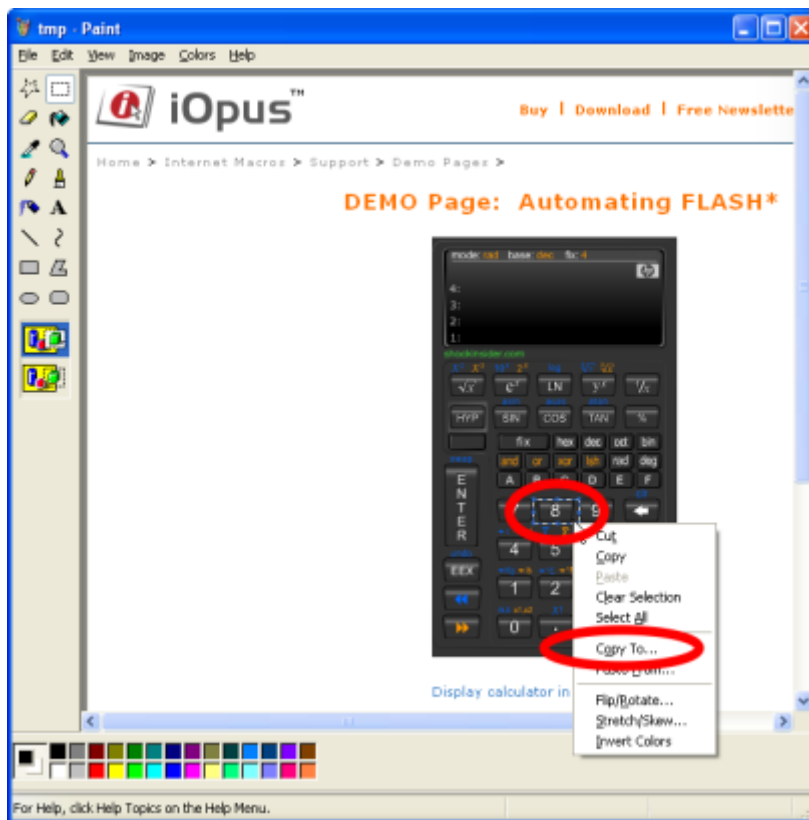
The following is a suggestion of a good and efficient method using the MSPAINT software. MSPAINT is a simple image editor that is part of the Windows operating system by default.

1. Surf to the website that you want to automate or test using the iMacro Browser.
2. Go to ImageSearch => Create ImageClip.

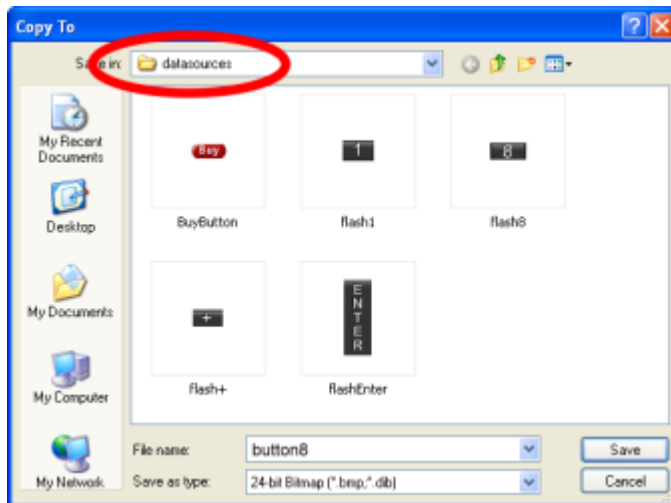
iMacros will then take a screenshot of the current web page and open it in MSPAINT:



3. In our example, we want to automate the Flash calculator, shown at <http://www.iopus.com/iim/demo/v4/flash/> and create an image for the button with the number 8.
4. In MSPAINT, choose the "Select" tool (see the red circle in the figure above).
5. Use the Select tool and mark an area around the button 8.



6. After you have marked the area around button 8, right-click the mouse and select the "Copy To..." command.



7. A file dialog box opens. By default, we recommend that you store the images in the datasources folder of iMacros.

8. Select a name for the selected area that you want to save, e.g. button8.bmp. You must store the image in "24-bit Bitmap" format.

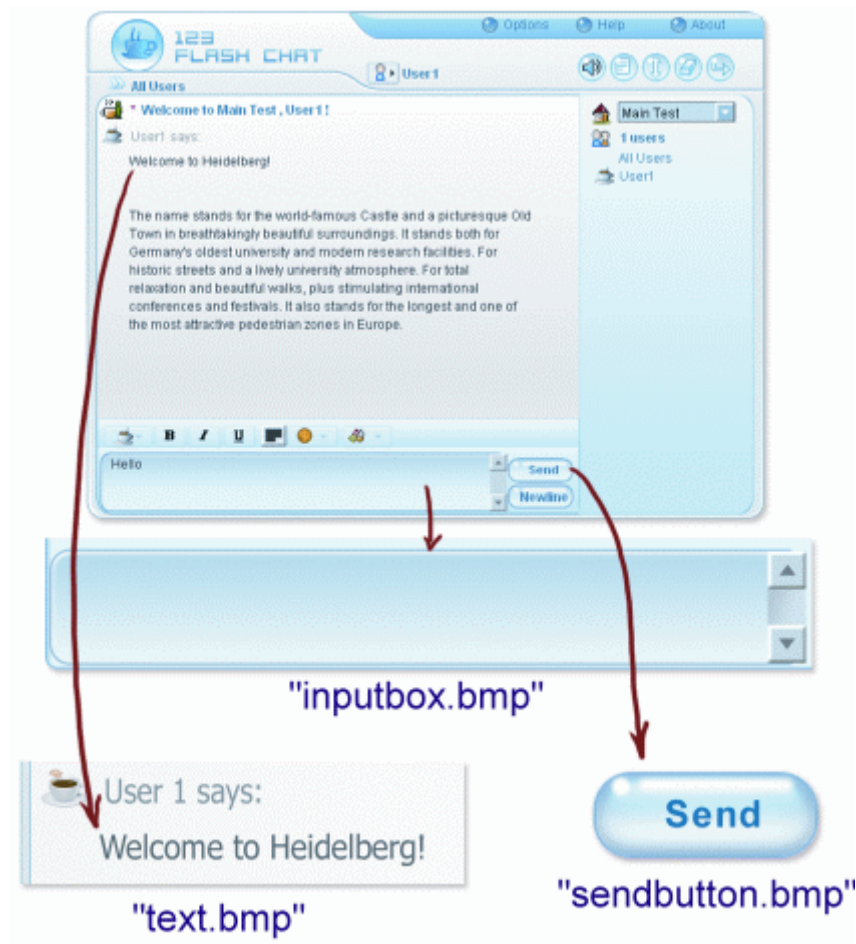
9. It's that easy; you are finished! You can now point the IMAGE parameter of any [IMAGESEARCH](#) ⁹⁶

or [IMAGECLICK](#)^[95] command to point to the just created button8.bmp

5.3 Example: Automate Flash Chat Web Applet

The diagram shows you that only three images and three lines of macro code are required to automate input into a flash (or Java) based chat applet. The picture below indicates that the procedure to [create images for IMAGESEARCH](#)^[61] has been done for the three elements.

This example includes a verification that the input is received.



The macro for this application is:

```
'Fill text into input box
IMAGECLICK[95] IMAGE=inputbox.bmp CONFIDENCE=95 CONTENT=Hello<SP>Heidelberg
'Click "send" button
IMAGECLICK[95] IMAGE=sendbutton.bmp CONFIDENCE=95
'Now test if the applet works => look for text "Heidelberg" in applet
window
IMAGESEARCH[96] IMAGE=text.bmp CONFIDENCE=95
```

If you add [STOPWATCH](#)^[109] commands to this macro, it can be used to [measure the performance](#)^[45] of the Flash (or Java) applet.

6 Distributing iMacros with your Application

The iMacros Scripting Edition contains a free iMacros Player license. This license allows you to distribute the iMacros Player royalty free with your application. The iMacros Player contains all features of the Scripting Edition, except the ability to record new macros.

Please contact the iOpus Support Team at <http://www.iopus.com/service/support/> to request your personal Player licence key.

There are two different approaches to distribute iMacros with your application.

Distributing the iOpus setup

You can distribute the normal setup executable from <http://www.iopus.com/download/>. When installing your application on the clients computer, you can easily call the iMacros setup and install the iMacros Trial version. You can then either write the Player key in the [im.ini](#)^[67] from your setup or let your client enter it manually. Your client will then be able to play any macro you created. The [Scripting Interface](#)^[53] also works. We recommend that you redistribute the original setup file. The setup supports some [command line switches](#)^[66] that make life easier.

Distribute iMacros with your setup

iMacros is a compact application that consists only of the following files, which you need to copy to your client's computer

| | |
|-------------------------------|--|
| <code>imacros.exe</code> | Mandatory, the iMacros engine |
| <code>imatl.dll</code> | Mandatory |
| <code>imsys.dll</code> | Mandatory |
| <code>iframe.exe</code> | Mandatory |
| <code>iimwnk.dll</code> | Mandatory |
| <code>iimConnector.dll</code> | Mandatory |
| <code>imgr.exe</code> | Mandatory, register it with <code>imgr.exe</code> /regserver after installation |
| <code>imi.ini</code> | Mandatory, settings file |
| <code>iimInterface.dll</code> | Optional, the Scripting Interface (COM object) |
| <code>iimRth.dll</code> | Optional, belongs to Scripting Interface |
| <code>imimage.dll</code> | Optional, the Image Recognition library |
| <code>imacros.dll</code> | Optional, the Internet Explorer Plug-in |
| <code>ab1.ico</code> | Optional, Icon for Internet Explorer Browser Bar |
| <code>ab2.ico</code> | Optional, Icon for Internet Explorer Browser Bar |

Depending on your version of Windows, you may also have to install and register the following redistributable Microsoft System Files into the Windows System directory. You will find them on any PC that has the regular iMacros installation:

`mscomctl.ocx`, `tabctl32.ocx`, `msstdfmt.dll`

The settings file `iim.ini` can be included in the setup for predefined settings, for example the iMacros Player License key. If you use this file, you need to store the location of the `iim.ini` file in

the registry:

```
Root: HKLM; Subkey: "Software\IIM"; ValueType: string; ValueName:
"AppDataFolder"; ValueData: "c:\myini\"
Root: HKLM; Subkey: "Software\IIM"; ValueType: dword; ValueName: "Use
AppDataFolder"; ValueData: "1"
```

6.1 Setup Command Line Parameters

The Setup programs (Installer and Uninstaller) accept optional command line parameters that assist you with the automatic deployment of iMacros. These can be useful to system administrators, and to other programs calling the Setup program.

Installer

The installer program accepts the following optional command line parameters.

```
/SILENT
/VERYSILENT
```

Instructs Setup to be silent or very silent. When Setup is silent, the Wizard and the background window are not displayed but the installation progress window is. When a setup is very silent this installation progress window is not displayed. Everything else is normal so for example error messages during installation are displayed.

If a restart is necessary and the `/NORESTART` command isn't used (see below) and Setup is silent, it will display a Reboot now? message box. If it's very silent it will reboot without asking.

```
/NOCANCEL
```

Prevents the user from cancelling during the installation process, by disabling the Cancel button and ignoring clicks on the close button. Useful along with `/SILENT`.

```
/NORESTART
```

Instructs Setup not to reboot even if it's necessary. Note that installing iMacros does *not* require a reboot unless iMacros was already installed *and running* on the machine during the setup.

```
/DIR="x:\dirname"
```

Overrides the default directory name displayed on the Select Destination Directory Wizard page. A fully qualified pathname must be specified. If the [Setup] section directive DisableDirPage was set to yes, this command line parameter is ignored. Example:

```
iopus-iim-setup.exe /VERYSILENT /DIR="c:\iim"
```

installs the software automatically in the `c:\iim` directory.

```
/GROUP="folder name"
```

Overrides the default folder name displayed on the Select Start Menu Folder wizard page. If the [Setup] section directive DisableProgramGroupPage was set to yes, this command line parameter is ignored.

```
/NOICONS
```

Instructs Setup to initially disable the "Don't create any icons" check box on the Select Start Menu Folder Wizard page.

Uninstaller

The uninstaller program (unins000.exe) also accepts an optional command line parameter.

/SILENT

When specified, the uninstaller will not ask the user any questions or display a message stating that uninstall is complete. Shared files that are no longer in use are deleted automatically without prompting. Any critical error messages will still be shown on the screen.

6.2 Restricted User Accounts

iMacros runs well on restricted user accounts, locked down systems and systems that use Active Directory services.

If you install the software for users with restricted rights, please note:

1. The software must be installed by a user with administrative rights
2. Restricted users that need to run the software must be given "modify" access to the following folder:

`C:\Documents and Settings\All Users\Application Data\iOpus-I-M`

iMacros uses the folder to store its settings and temporary data. **The standard Windows 2000/XP/2003 restricted users have the correct permissions for this folder by default.**

Alternatively you can change the location of the settings folder in the User Settings tab of the Options dialog. This can be especially useful in an Active Directory environment.

6.3 Modify settings directly

All settings used by iMacros are stored in a text file called `imi.ini`. We do not recommend editing this file manually unless you know what you are doing. If in doubt, please contact the iMacros Support Team. The entries in this file are saved in the standard user setting way, i.e. in key=value pairs. The keys should be self-explanatory.

In Windows 2000, XP and 2003, this file is typically located in `C:\Documents and Settings\All Users\Application Data\iOpus-I-M`.

Since this is a system directory, it is "hidden" by default. In most file managers (e.g. the Microsoft Windows Explorer) you need to enable the "display hidden / system files" option (or similar function) in order to see this directory. In the Microsoft Windows Explorer this option is located at View -> Folder Options -> View Tab -> Go to the "Hidden Files" option and select "Show all".

In Windows 98 and ME, the file is located in the Program Folder, which is typically `C:\Program Files\InternetMacros`.

How to change settings in this file:

1. Stop & close iOpus iMacros.

2. Locate `imi.ini`.
3. Open `imi.ini` with a text editor (e.g. Notepad) and make the changes. Then save the file under the same name.
4. Restart iMacros
5. The new settings are active.

7 Frequently Asked Questions

We separated the Frequently Asked Questions we got over the years into [Getting started](#)^[68], [Installation](#)^[69], [Creating macros](#)^[70], [Data extraction](#)^[75] and [Web testing](#)^[79] related questions. Some topics in this section are duplicated topics that proved to be hard to find in the online help and the user manual, while others are solutions for a specific task.

7.1 Getting started

7.1.1 The web page I am accessing requires IE. Is this a problem?

Q: The web page I am accessing requires Internet Explorer (IE). Is this a problem?

A: The iMacros browser has full IE compatibility. It comes with all Internet Explorer functions, including Cookies, ActiveX Controls, Java Script and Macromedia Flash. When you are using the iMacro Browser you can also set the useragent string using the command line switch [-useragent](#)^[137].

7.1.2 Are there conditional statements in the Internet Macros language?

Q: Are there conditional statements like `if... then... else` in the iMacros macro language?

A: The iMacros language is designed as a descriptive language (similar to HTML) and does not contain conditional statements. We did not add such statements as we do not think that our customers should have to attend week-long seminars just to learn yet another proprietary scripting or programming language.

Instead, we added a command line interface for use with [batch files](#)^[50] and [task scheduler](#)^[52]. We also added the powerful [Scripting Interface](#)^[53] that allows you to use iMacros with every Windows Scripting or programming language on the planet. Examples are [VBS](#), VB, VBA, VB.NET, [Perl](#), [Java](#), [Foxpro](#), C, C++, C# and many more. These languages are used by millions of computer users, are reliable and very well documented.

Examples for VBS can be found [here](#)^[54] and on the [iOpus homepage](#).

7.2 Installation

7.2.1 I create macros for my clients. Do you have a free player?

Q: I create macros for my clients. Do you have a free player?

A: Yes, the Scripting Edition includes an iMacros Player license key. This license allows you to ship macros and programs based on iMacros along with a copy of iMacros itself to your users *without* royalty payment. Technically speaking, this means that if iMacros is unlocked with the iMacros Player license key, your users have all the features of the Scripting Edition available except the ability to record new macros. There exist several options on how to redistribute the [iMacros files](#)^[65].

Please contact the iOpus Support Team to request your Player licence key.

7.2.2 How can I automatically install iMacros?

Q: How can I automatically install iMacros?

A: General information can be found [here](#)^[65]. But, basically there are two options:

(1) If you want to distribute the files within your setup, please [contact us](#) for a list of required files and an installation script source code. You can then modify this installer for your own needs or simply use it to learn how the software is installed.

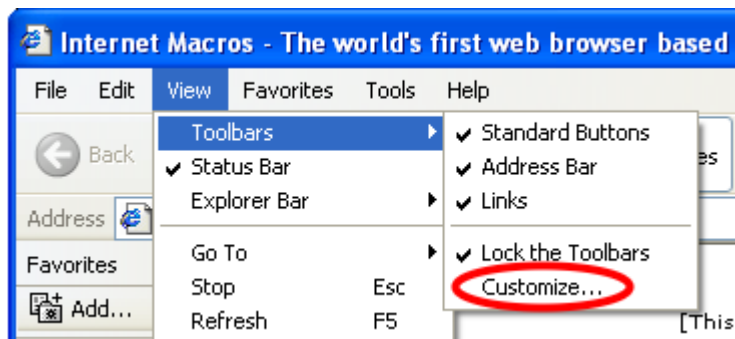
(2) You can use the original setup file, using [command line switches](#)^[66] to control the behaviour of the installer.

7.2.3 What if the iMacros icon does not appear in the IE toolbar?

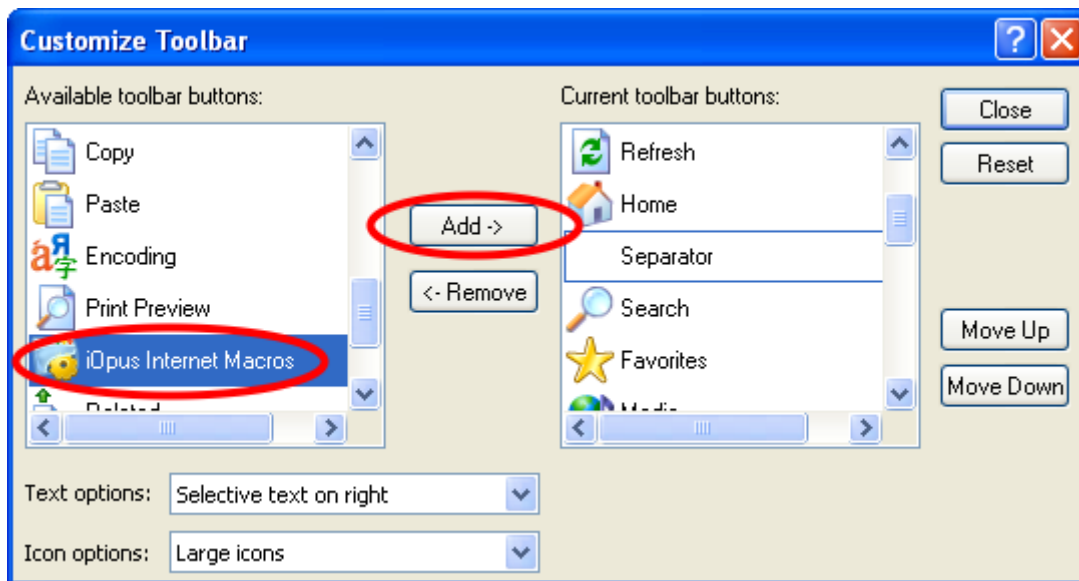
Q: What if the iMacros icon does not appear in the Internet Explorer toolbar?

A: If you have previously customized your Internet Explorer toolbar, the new iMacros icon does not appear by default. You need to add it to the Internet Explorer toolbar manually:

1. To add the icon, open your Internet Explorer and in the menu select View > Toolbars> Customize



2. Select the iMacros icon from the list of "Available Toolbar Buttons" and add it.



3. Done! The icon appears in the toolbar.

Note: This behavior is a feature of Internet Explorer - and not "bug" with the iMacros software. Microsoft's philosophy behind this seems to be that users who customized their browser toolbar once, do not want that new icons are added automatically.

7.3 Creating Macros

7.3.1 How to...

7.3.1.1 How to automate pages where links (URLs) change every time I visit the page?

Q: A link changes every time I visit the web page ("session ID"), how can I replay the macro without error?

A: Some web sites create links with a "session id" that is different each time you log in. What you need to do is to edit the macro manually after you recorded it and replace the id (or any other changing part of the string) with the wildcard *.

This method is described in detail in [Fine Tune TAG Commands](#)^[24] or [Extract Tech Tip](#)^[40].

7.3.1.2 How can I print a selected frame?

Q: How can I print a selected frame?

A: If the page uses frames and you want to print only a specific frame, select this frame with [WINCLICK](#)^[42] first before you use [PRINT](#)^[103]. An example macro can be found in the [PRINT chapter](#)^[21] of this manual.

7.3.1.3 How do I set the focus to a input field for manual entry?

Q: Some websites require me to enter a graphical "security code" or Turing number. I want the login macro to stop at this point and enter the characters manually. How do I set the focus to a input field for manual entry?

A: To set the focus to a certain input field (or any other element on the screen), please use the [WINCLICK](#)^[115] command.

7.3.1.4 How to create macros that will run on the page displayed in the web browser?

Q: Is it possible to create macros that will run on the page displayed in the web browser, rather than using the [URL](#)^[113] GOTO command to direct it to a web page at the beginning of a macro. The pages I want to run macros on are all set out identically so one macro should work on all the pages.

A: The URL command is inserted by default every time you start recording a macro. After recording a macro, edit the macro source and manually remove the [URL](#)^[113] GOTO command. If you want to run the same macro on different pages, yet not all pages are exactly the same, use [SET](#)^[107] [!ERRORIGNORE](#)^[121] YES to simply skip [TAG](#)^[110] commands that cannot be executed.

7.3.1.5 How to read and write from a database?

Q: How can I read and write from a database?

A: Database integration is very easy with iMacros thanks to its powerful [Scripting Interface](#)^[53]. iMacros can connect to any Windows database, for example Access or SQL Server, but also MySQL. The Scripting Interface can be used from any Windows programming or scripting language like Excel VBA, Foxpro, Perl, C, C++, Java with no problems.

Please see the following example scripts:

"extract-2-file.vbs" (uses wsh-extract-and-fill-part1 and wsh-extract-and-fill-part1 macros)

"extract-2-database.vbs" (store information from a website in a database)

"file-2-web.vbs", "database-2-web.vbs" (reads a text file or database and submits the content to a website)

7.3.1.6 How do I make the macro continue (and not stop), if somewhere in the macro I get a timeout or error?

Q: How do I make the macro continue (and not stop), if somewhere in the macro I get a timeout or error?

A: Please add the command [SET](#)^[107] [!ERRORIGNORE](#)^[121] YES to your macro.

7.3.1.7 How can I do calculations in a macro?

Q: How can I do calculations in a macro?

A: For very simple calculations, iMacros offers the [ADD](#)^[87] command.

But instead of doing calculations in the macro, you can perform the calculations inside a script (e.g. Visual Basic Scripting, VBS). This allows you to use all kinds of more sophisticated calculations, loops (e.g. for....next loops) or conditional logic (e.g. if/then) for automating your web tasks. You can then use [iimSet](#)^[144] of the [Scripting Interface](#)^[139] to send values to a macro and the [EXTRACT](#)^[38] command

to return values **from** a macro to the script.

The file `random-numbers.vbs` demonstrates how to use a script to generate random numbers and modify the macro based on the outcome. Or, see the file `extract-and-fill.vbs` on how to extract a value, subtract a number from it and re-submit the new value to the website.

Note that VBS files are not started from within the iMacros Browser. You can locate the file (e.g. via Explorer) and then simply double-click on the VBS file. Windows will then start to process the VBS file. The iMacros Browser is started by the VBS file via the `iimInit`^[142] command automatically.

More information about VBS files can be found in our online [Windows Scripting Host tutorial](#) or [in this manual](#)^[53].

7.3.1.8 How do I link several macros together?

Q: How do I link several macros together or run one after the other?

A: There are several method to combine different macros.

Copy & Paste

iMacros are simple text files, so you can easily combine (merge) several macros by editing the text files. The `.iim` macro files can be viewed and edited with an editor (e.g. Notepad), see [here](#)^[12].

Batch files

If all you need is to run several macros, one after the other, you can call them using the command line interface in a [batch file](#)^[50]. The `-noexit`^[135] command line switch may prove helpful in this context. Example batch files can be found in the `Examples/Batch Files` folder of your iMacros installation.

Scripting Interface

You can use the `iimPlay`^[143] command to run one macro after the other. The following macro continues *exactly* at the same position where the other macro stopped as the browser window remains open between each macro. Remember to remove the `URL GOTO...`^[71] command from the following macros, *if* you want it to continue exactly where the other macro left off. To close a browser window, you can use `iimExit`^[140]. The documented example script `combine-macros.vbs` can be found in the `Examples/Windows Scripting Host` folder of your iMacros installation.

7.3.1.9 How to make iMacros stop until a users enters a value?

Q: If someone needs to make that decision meaning that the task would be partially automated, is there a way to make iMacros stop until a users enters a value and continue afterwards?

A: You can add the `PROMPT`^[103] command to the macro to make iMacros ask for a variable (e.g. the file name) or simply wait for a click on OK.

7.3.1.10 How to create a macro that can select one from a series of radio buttons?

Q: How can I create a macro that can select one from a series of radio buttons? I want to avoid setting up separate macros for each selection.

A: You can use variables to select a radio button. For example, instead of using one of these three

lines:

```
TAG POS=1 TYPE=INPUT:RADIO FORM=ACTION:/cgi-bin/Info.cgi ATTR=NAME:lunch
CONTENT=yes
TAG POS=1 TYPE=INPUT:RADIO FORM=ACTION:/cgi-bin/Info.cgi ATTR=NAME:lunch
CONTENT=no
TAG POS=1 TYPE=INPUT:RADIO FORM=ACTION:/cgi-bin/Info.cgi ATTR=NAME:lunch
CONTENT=maybe
```

You can use:

```
TAG POS=1 TYPE=INPUT:RADIO FORM=ACTION:/cgi-bin/Info.cgi ATTR=NAME:lunch
CONTENT={{choice}}
```

and set {{choice}} to either "yes", "no" or "maybe". Please note that this value might be case sensitive. This depends on the website that processes this input.

To assign a value to the variable, you can use either the [SET](#)^[107] command, the `-var_varname varvalue` [command line](#)^[137] argument or the [iimSet](#)^[144] command of the [Scripting Interface](#)^[139].

7.3.1.11 How to display the content of a variable?

Q: How to display the content of a variable?

A: For testing and debugging macros it can be useful to display the content of some variables. You can divert the [PROMPT](#)^[103] command from its intended use for this purpose. If you want to display the value of the two variables, !COL2 and MY_VARIABLE, use the following command

```
PROMPT {{!COL2}} {{!VAR1}} {{MY_VARIABLE}}
```

This will open an input dialog for !VAR1. This is the main purpose of a PROMPT command, but for our current use of the command this can be ignored. The input dialog box displays the value of {{!COL2}} as dialog box text and the value of {{MY_VARIABLE}} as input box default value. This allows you to check if these variables are populated correctly.

7.3.2 Issues during Replay

7.3.2.1 Does iMacros work with every web site?

Q: Does iMacros work with every web site?

A: The macros recorded with iMacros work on almost all sites. Even if the standard click methods fail (for example on Flash or Java applets) the [WINCLICK](#)^[42] command can be used in almost all cases. If you think you have found a web site where iMacros does not work [please let us know](#). We are very interested to hear of such cases and will assist you in finding a solution.

7.3.2.2 Can we have a loop inside a macro?

Q: We have a macro that goes to a web site and issues a query. The result is a variable list of links that all need to be clicked and the result saved for each link. Can this be accomplished using the

tool?

A: Basically what you need is a macro that first navigates to your website, runs the query and then loops over the result. The solution is to split the task in two macros. The first macro (macro1) navigates to the site and runs the query. The second macro (macro2) "loops" over the result until all items are processed. Both macros are combined with a small script. Note that you have to manually remove the `URL GOTO...` in macro2. Thus the iMacros Browser continues macro2 exactly at the position where macro1 stopped. As an example, please see the "combine-macros.vbs" example script.

7.3.2.3 Does the macro script wait for the page to fully finish loading?

Q: Does the macro script wait for the page to fully finish loading?

A: Yes, if a command inside a macro triggers a page (re-)load, iMacros automatically waits until the page is completely loaded before it continues with the next command.

On a few very complex web pages with frames, the macro may continue before everything is loaded completely. In this case you can tell iMacros manually to wait for another "Page loaded..." signal by adding `WAIT[114] SECONDS=#DOWNLOADCOMPLETE#` command to the macro.

Note: If you use the [Image Recognition Plugin^{\[60\]}](#) you can also search for an image using the `IMAGESEARCH[96]` command. iMacros will wait until this image appears (or the timeout value is reached). This approach looks at the website as if it were an *image* - just as a human would do! Thus this approach works with every web site, regardless of which web technology is used (HTML, Frames, ASP.NET, Java applet, Flash applet,...).

7.3.2.4 Why is a certain input box never recorded?

Q: Why is a certain input box never recorded?

A1: If all input boxes except one or two are recorded correctly, the problem could be that these boxes have the same internal name. Typically, this occurs with input boxes that are used for "From" and "To" or "Password" and "Re-enter password" type of inputs. If (and only if) both fields have the same internal name iMacros can not fill the fields correctly during replay.

Solution: Manually edit the macro and use `POS=2` to reach the second element - the "To" selection in the following example.

```
URL GOTO=http://www.flybmi.com/bmi/en-gb/index.aspx
SIZE X=800 Y=600
'Fill FROM field
TAG POS=1 TYPE=SELECT FORM=NAME:IBEMagnet_IBEMagnet__ctl0 ATTR=NAME:
CONTENT=$*Dublin*
'Fill TO field
TAG POS=2 TYPE=SELECT FORM=NAME:IBEMagnet_IBEMagnet__ctl0 ATTR=NAME:
CONTENT=$*Frankfurt*
TAG POS=1 TYPE=INPUT:IMAGE FORM=NAME:IBEMagnet_IBEMagnet__ctl0
ATTR=NAME:&&VALUE:
```

A2: If you click on input boxes and no `TAG[110]` commands at all are created, you might have encountered a Flash- or Java-based input box (inside a Flash- or Java-Applet) instead of a standard HTML input box. In this case, change the ClickMode to `WinClick[42]` to record the input.

7.3.2.5 Why is dialog XYZ not handled by iMacros?

Q: Why is the dialog "XYZ" not handled by iMacros? [XYZ can be any browser dialog, for example the [print](#)^[20], [download](#)^[16] or [save](#)^[14] dialog]

A: This issue can have several causes: Here are some questions in order to diagnose this problem:

Dialog is not recognized during Recording

- During recording, an iMacros dialog should appear under the Web Browser dialog. If the iMacros dialog does *not* appear, please make sure that the option "Use IM Dialog Manager" is checked in the Browser tab of the Options dialog.
- If this option is checked and the dialog is still not recognized, please play the "Demo-Javascript", "Demo-Download" or "Demo-SaveAs" macro. Are the dialogs in these macros handled correctly? If not, try reinstalling the software or contact tech support.
- Does the dialog appear before you start recording? This could happen, for example, with a [login dialog](#)^[18]. To work around this, start recording at a different page, and enter your real web site address. You can always edit the macro after recording and remove the first URL.
- If the dialog is still not recognized: Is the dialog in our list of [supported dialogs](#)^[18]? If yes, try adding the commands manually to the macro and test if everything works correctly during replay.

2. Dialog is recorded, but not managed during Replay

- If the dialog appears at the end of the macro, it can be that the macro stops shortly before the dialog appears and thus cannot "catch" it. To work around this, add a [WAIT](#)^[114] SECONDS=3 command after the command that triggers the dialog to make sure iMacros is still active when the dialog appears.

7.3.2.6 How do I make the iMacros Browser appear as native IE (Internet Explorer)?

Q: How do I make the IM Browser appear as native IE (Internet Explorer)?

A: Open the iMacros IE Plug-in or the iMacros Browser and open the Options dialog. In the Options dialog, select the Browser tab and choose "Stealth mode (Identify as native Internet Explorer)".

7.4 Extracting Data

7.4.1 I use the EXTRACT command and get the message "Extraction anchor not found". What is the solution?

Q: I use the [EXTRACT](#)^[91] command and get the message "Extraction anchor not found". What is the solution?

A: The reason for this message is, that the extraction anchor, given mainly by the ATTR parameter of the [EXTRACT](#)^[91] command could not be found on the currently active web site. Some tips on creating good extraction anchors:


Make sure that you selected the appropriate part for the anchor. It should only contain HTML code that does not change during each session or the changing parts need to be replaced by the wildcard *.

Incorrect: *This anchor does not work because it contains a changing value (the dollar amount).*

```
EXTRACT POS=12 TYPE=TEXT ATTR=<TD>Price US$ 3.50</TD><TD>*
```

Correct: *Replaced changing part with asterisk **

```
EXTRACT POS=12 TYPE=TEXT ATTR=<TD>* </TD><TD>*
```

To test your anchor, please make sure you click on  after iMacros suggests an anchor element or you have manually created one. Most extraction issues are detected with the test feature.

Experiment with the `POS` attribute. `POS` indicates how often the anchor is found on a web page before the extraction is started.

If the information you want to extract is inside a framed web site, you need to click inside the frame that contains the information you want to extract *before* opening the Extraction Wizard. This generates the `FRAME`^[94] command and marks the frame as active for the extraction.

7.4.2 How can I insert the extracted information back into same webpage?

Q: I require macros that can extract simple text data from a web page and then insert the extracted information back into either the same web page or another web page. I do not need to store this data in any way. In effect all I need to do is copy and paste the text data.

A: Extracted data is saved in the `!EXTRACT`^[122] variable. You can assign the extracted value to a variable and use it again at a later point in your macro.

```
EXTRACT POS=29 TYPE=TEXT ATTR=<TD>*
SET !VAR1 { { !EXTRACT } }
...
TAG POS=1 TYPE=INPUT:TEXT FORM=NAME:f1 ATTR=NAME:n1 CONTENT={ { !VAR1 } }
```

7.4.3 How to extract information from a table with variable length and/or more than one page?

Q: How do I extract information from a table with variable length and/or more than one page?

A: This task occurs often, e.g. when you extract data from a database (search engines, flight schedules, news service). Depending on your keyword, you might get zero, one, two or 500 entries back. To extract this data, need to divide your macro into two separate macros, as described [here](#)^[73]. Your second macro only consists of one extraction tag for one line of the table, e.g.:

```
EXTRACT POS={ { mypos } } TYPE=TEXT ATTR=<TD<SP>noWrap>*EUR*
```

After replaying the first macro, which navigates to the results, you need to loop through them by changing the `POS` parameter, i.e. the `mypos` variable from 1 to 999. This line would be the content of the "macro_extract" macro in the source code sample below. It extracts the EUR price on each line.

Once you get the `#EANF#` message back instead of a result, you know that you reached the end of the table on the current page.

If necessary, you can now run another macro to click on a "NEXT" link for another page of results. If no "NEXT" link is found, you know that you are done.

Here is a Visual Basic source code snippet, that shows how the different macros are used:

```
'Read keywords
For i1 = 1 To 999 '<<<<<<<<< LOOP Keywords until all words processed (=>
Read_Line returns "ERROR")
    sKeyword = Read_Line(mFileInput, i1) 'Read keyword from a text file
    If sKeyword = "ERROR" Or Len(sKeyword) < 2 Then
        Exit For
    End If
    iRet = iiml.iimDisplay("Search: " + sKeyword)
    iRet = iiml.iimSet("-var_search", sKeyword)
    iRet = iiml.iimPlay("macro_search")
    sData = iiml.iimGetLastExtract()

    iRet = iiml.iimDisplay("Extract: " + sKeyword)

    For i2 = 1 To 999 '<<<<<<<<<<<<<<<LOOP "next" links
        'Loop all NEXT links until error => no more NEXT links to process)

        For i3 = 1 To 999 '<<<<<<<<<<<<<<< LOOP table lines and extract
            them
                'Loop the table rows until an error occurs => no more data on this
                page

                DoEvents
                iRet = iiml.iimSet("-var_mypos", CStr(i3))
                iRet = iiml.iimDisplay(sKeyword + "P:" + CStr(i2) + "L:" +
                CStr(i3))

                iRet = iiml.iimPlay("macro_extract")
                sData = iiml.iimGetLastExtract()
                If iRet = 1 and len (sData) > 0 Then
                    'Data found, now split it and save it to a file
                    sSplit = Split(sData, "[EXTRACT]")
                    s0 = sSplit(0)
                    If UBound(sSplit) > 0 Then s1 = sSplit(1)
                    If UBound(sSplit) > 1 Then s2 = sSplit(2)
                    i = InStr(s0, "#EANF#")
                    If i <= 0 Then
                        sLine = sKeyword + sSep + CStr(i2) + sSep + CStr(i3) +
                        sSep + s0 + sSep + s1 + sSep + s2
                        Call Write_Line(mFileOutput, sLine)
                    Else
                        Exit For 'next page
                    End If
                Else
                    Exit For 'next page
                End If 'iRet
            Next 'table rows loop

            iRet = iiml.iimDisplay(sKeyword + " Page: " + CStr(i2))
            iRet = iiml.iimPlay("macro_next")
            If iRet < 0 Then
                Exit For 'next keyword
            End If

        Next 'NEXT links loop
```

Next 'Keyword list loop

```
iRet = iiml.iimDisplay("Extraction completed")
```

Note: The support functions used in this example are:

Function Read_Line(sFile, iline) As String

```
Dim sValue
```

```
Dim i As Integer
```

```
Dim bFound As Boolean
```

```
bFound = False
```

```
Open sFile For Input As #1
```

```
i = 1
```

```
Do While (Not EOF(1) And bFound = False)
```

```
    Input #1, sValue
```

```
    If i = iline Then bFound = True
```

```
    i = i + 1
```

```
Loop
```

```
Close #1
```

```
If bFound = False Then
```

```
    sValue = "ERROR"
```

```
End If
```

```
Read_Line = sValue
```

```
End Function
```

Public Sub Write_Line(sFile, sLine)

```
    Open sFile For Append As #2
```

```
    Print #2, sLine
```

```
    Close #2
```

```
End Sub
```

7.4.4 How to work (fill/extract) with hidden input fields?

Q: How do I fill data to or extract data from a hidden input field?

A: Sometimes web forms contain internal data that is passed between pages in hidden input fields. Since you can not see these fields, you can not record filling or extracting visually, but you can edit the recorded macro afterwards. How this is done is described [here](#)^[110] (fill) and [here](#)^[91] (extract).

7.4.5 How do I extract text from message boxes?

Q: How do I extract text from a (Javascript) message box?

A: You only need to add:

```
SET !EXTRACTDIALOG[123] YES
```

to your macro and the text of a message box is extracted. The text is processed the same way as standard [extracted](#)^[30] text. Examples of Javascript dialogs can be found in the demo macros "Demo-JavascriptDialog" and "Demo-OfflineDialogs".

7.4.6 How do I extract data separated with
 ?

Q: Sometimes, multiple items inside one cell are separated by
 instead of creating individual cells for the data. How do I extract data separated with
?

A: To work around this problem, use the TYPE=HTM extraction. It preserves all HTML tags inside the text so that the extraction result can be separated later. See also the "[Demo-Extract](#)^[36]" example macro for more details.

7.4.7 How do I extract the page URL?

Q: How do I extract the page URL?

A: The URL of the current page is stored in the internal variable `{{!URLCURRENT}}`^[130]. You can use this value inside of commands like any other variable. Or extract it by assigning the value to the [EXTRACT](#)^[91] command:

```
SET !EXTRACTADD {{!URLCURRENT}}
```

7.5 Web Testing

7.5.1 How can I search for a specific keyword on a web page?

Q: How can I search for a specific keyword on a web page? When parsing a web site you want to find out if a certain word exists on a web page to trigger further action (like saving the web page, printing the page or running the next macro).

A: There are many applications of the keyword search feature. For example tracking your search engine ranking, observing a (financial) news web site, get feedback on a failed login, watching web pages for changes or even monitoring trademark infringements automatically.

iMacros offers three options for keyword search (keyword assert):

1. [TAG command](#)^[79]
2. [IMAGESEARCH command](#)^[81]
3. [EXTRACT command](#)^[80]

Please click the links for more details on each option. We also created the "Demo-Keyword-Check.iim" example macro.

7.5.1.1 Keyword Search with TAG

The [TAG](#)^[110] command searches for a HTML tag inside the web page. If found, it clicks on it. But [TAG](#)^[110] can also search for any kind of text on a web page. Thus, any [TAG](#)^[110] command is automatically a keyword assert command. For example

```
TAG POS=1 TYPE=STRONG ATTR=TXT:*iMacros*
```

searches for the word iMacros enclosed by the STRONG HTML tag. If you want to search for text regardless of the format, i.e. the enclosing HTML tag, please use the wildcard character * instead of a HTML tag:

```
TAG POS=1 TYPE=* ATTR=TXT:*iMacros*
```

This command is the same as above, but this time the command ignores the formatting of the keyword. This command takes longer, as iMacros needs to scan the complete page, not only certain

html tags.

You can also "invert" the keyword search and generate an error if the keyword *is* found on the page:

```
TAG POS=1 TYPE=* ATTR=TEXT:*iMacros* CONTENT=EVENT:FAIL_IF_FOUND
```

As the event suggests, the [TAG](#)^[110] command will now generate an error if it succeeds.

7.5.1.2 Keyword Search with EXTRACT

You can use the [EXTRACT](#)^[91] command and make the keyword the extraction anchor. The general idea is to try and extract a certain piece of information and then check if the extraction result is the Extraction Anchor Not Found (#EANF#) message.

Example: We want to find out if the words "Order completed" are displayed on a web page. If yes, we want to print the page. To search the web page for the test phrase, create a macro called `mysearch`, which only has two lines:

```
VERSION BUILD=4210125
EXTRACT POS=1 TYPE=TEXT ATTR=*Order<SP>completed*
```

In this example, we are searching the web page for the first occurrence (POS=1) of the keyword "Order completed". If the message #EANF# is returned, then the keyword was not found as the keyword *is* the data extraction anchor. If the keyword was found, the [EXTRACT](#)^[91] command returns the complete text of the found HTML tag. In our example, this could be "Software Order completed".

To print the web page, create a macro called `print_this`. It has only two lines:

```
VERSION BUILD=4210125
PRINT
```

To connect both macros together, create a small Windows script:

```
set iiml= CreateObject ("InternetMacros.iim")
iret = iiml.iimInit()
iplay = iiml.iimPlay("mysearch")
extracted_text = iiml.iimGetLastExtract()

'test if keyword appeared on website.
If iplay = 1 Then
    If instr (extracted_text, "#EANF#") > 0 Then
        MsgBox ("Sorry, keywords not found")
    Else
        iplay = iiml.iimPlay("print_this")
    End If
End If

If iplay < 0 Then
    MsgBox "Error!"
End If
```

Note: The same procedure can be used to look for several keywords on a page, for example "cat", "dog" and "mouse":

The solution is to use several [EXTRACT](#)^[91] commands. So in the macro use:

```
EXTRACT POS=1 TYPE=TXT ATTR=*cat*
EXTRACT POS=1 TYPE=TXT ATTR=*dog*
EXTRACT POS=1 TYPE=TXT ATTR=*mouse*
```

In the script, you can look at each array element to see if they keyword was found:

```
iplay = iiml.iimPlay("wsh-extract-rate")
data = iiml.iimGetLastExtract()
If iplay = 1 Then
    data= Split(data, "[EXTRACT]")
    If data(1) <> "#EANF#" Then MsgBox "Keyword CAT found!"
    If data(2) <> "#EANF#" Then MsgBox "Keyword DOG found!"
    If data(3) <> "#EANF#" Then MsgBox "Keyword MOUSE found!"
End If
```

7.5.1.3 Keyword Search with IMAGESEARCH

You can use the [IMAGESEARCH](#)^[96] command to look for keywords or keyimages. Thus, instead of a text you supply an image of the word:

Example (Image of the word "iMacros"): **iMacros**

The advantage of this command is that it works on *any* type of web page, including Flash, Java or ActiveX controls.

For more details please see the [Image Recognition Plugin](#)^[60] chapter.

7.5.2 How to test that certain images will show up when a page is loaded?

Q: I want to test that certain images will show up on my site when a page is loaded.

A: You can use the [Save Item](#)^[16] function for this. If a image is missing, an error will occur. Example:

```
CLEAR
URL GOTO=http://www.iopus.com/iim/demo/v4/images/test-image-loaded
TAG POS=1 TYPE=IMG
ATTR=TEXT:<IMG<SP>height=90<SP>src="bee.jpg"<SP>width=90<SP>border=0>
CONTENT=EVENT:SAVEITEM
TAG POS=1 TYPE=IMG
ATTR=TEXT:<IMG<SP>height=90<SP>src="shark_thumbnail.jpg"<SP>width=90<SP>border=0>
CONTENT=EVENT:SAVEITEM
```

This macro will download two images. If one image is missing on the web page, the macro stops with an "image not found" error. To make sure that the missing image is not stored in the browser cache, it is recommended that you use the `CLEAR` command before this test.

This technique is also recommended if you are automating or testing web sites with complex frames. To make sure a specific frame is completely loaded, do the above test for a specific image in this frame. iMacros will wait at the `TAG EVENT:SAVEITEM` command until this image appears.

The above method is easily implemented if you have information about the image such as its own URL, the link URL or the alternative text. If all you have is the image itself, use the [IMAGESEARCH](#)^[96] command of the [Image Recognition Plugin](#)^[60].

7.5.3 What effects has the iMacros Browser itself on application response measurements?

Q: What effects does iMacros itself have on web application response measurements?

A: The overhead of iMacros itself is very thin. On a 3 GHz Pentium, each macro step takes only about 20 ms (0.02s) of processing time. If you compare this to the average page response time, the delay caused by iMacros itself is typically much less than 2%. Note that a few commands need more processor time, for example the [CLEAR](#)^[88], [IMAGESEARCH](#)^[96] or [IMAGECLICK](#)^[95] commands.

Since the iMacros Browser is an IE-compatible browser, realistic viewer experience results are guaranteed. For example, your performance measurement results include the browser rendering time.

7.5.4 How to set up a 7x24h (non-stop) operation?

Q: How can I set up a 24 hours a day, 7 days a week (non-stop) operation with iOpus iMacros? .

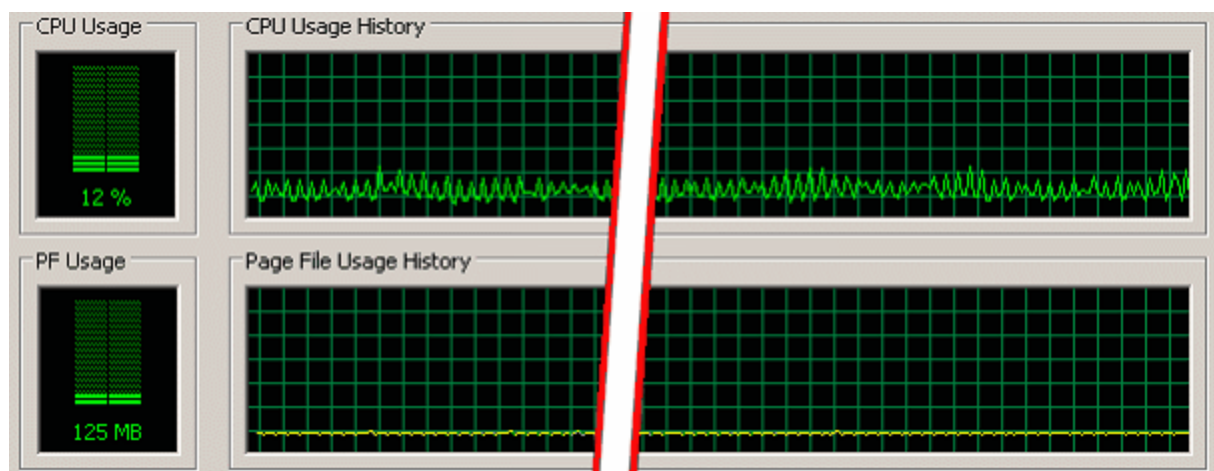
A: Many applications of iOpus iMacros require continuous operation of the software. Examples are the use of IIM as part of payment gateway, extracting large volumes of information or web testing applications in general.

Problem:

iMacros has the ability to remotely control the web browser and a wide variety of third-party browser plug-ins, such as Java virtual machine (Java applets), Adobe Acrobat Reader (PDF viewer), Macromedia Shockwave (Flash applets) and many others. By design most of these controls are not intended for 7x24 operations and running them repeatedly for a several days can lead to undesirable effects, such as increased memory consumption ("memory leaks").

Solution:

The Scripting Interface is a compact and reliable component that controls the iMacros browser and, thus, all its plug-ins. The Scripting Interface was designed to automate web testing and at the same time to compensate problems of the Windows Internet components used by the iMacros browser and third-party browser plug-ins.



Picture 1: Stable CPU and memory usage over time running the sample code listed below in part (b) with the "[Demo-Flash](#)^[42]" macro. The left part is a task manager snapshot taken on day 1 and the right part is taken on day 30. The snapshots were taken on a Windows Server 2003 with 2 GHz CPU and 512 MB Ram.

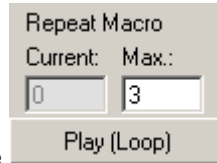
Machine Restart

A periodic restart is *not* required. However, as with server application in general, you should make sure that the application restarts correctly after the machine is rebooted or reset.

If you design an application for 7x24 operations we recommend the following

Use the Scripting Interface instead of the Play (Loop) Button

If you need to repeat a certain macro for example five million times, do not start a single browser



instance and use the [Scripting Interface](#)^[139] for the loop. In [VBS code](#) this looks like:

```
Set iim1= CreateObject ("InternetMacros.iim")
For m = 1 to 5000000
    iret = iim1.iimInit()
    iret = iim1.iimPlay("macro1")
    iret = iim1.iimExit()
Next
```

This command sequence closes the browser after each macro and thus avoids browser memory leaks. Even if the browser or one of its plug-ins "hangs" or "crashes", the Scripting Interface is *not* influenced by this and will open a fresh browser instance and continue to work. However, closing and re-opening the browser (as the above example does) takes a few seconds on an average PC. So if processing speed is important for your application, you can split this task in two loops, and use iimExit only, for example, every 1000th loop to save time.

```
Set iim1= CreateObject ("InternetMacros.iim")
For m = 1 to 5000
    iret = iim1.iimInit()
    For n = 1 to 1000
        iret = iim1.iimPlay ("macro1")
    Next
    iret = iim1.iimExit()
Next
```

8 iMacros Commands Reference

Macro Commands

This page lists all available iMacros commands. Each command has either zero or more parameters. If parameters can be omitted, they are enclosed by square brackets. If several choices are possible for the same parameter, they appear in brackets and are separated by the | character. Integer numbers are denoted by the letters *n* or *m*, all other name denote a series of characters (strings). For Scripting Interface commands please look [here](#)^[139].

^[91]

The ' character indicates a comment. If a line starts with ' everything behind the ' is ignored. Typically this is used for comments or to disable specific parts of a macro.

Note: **A macro cannot have empty lines**, as an empty line indicates the end of the macro. So every line in the macro must have at least the comment symbol.

ADD^[87] *result_var added_value*

Adds a value to a variable.

BACK^[88]

Opens the previously visited webpage.

CLEAR^[88]

Clears browser cache and cookies on the harddrive.

CLICK^[89] *X=n Y=m [CONTENT=some_content]*

"Clicks" on the element at the specified X/Y coordinates.

CMDLINE^[90] *variable default_value*

Sets the variable to a value retrieved from the command line.

DISCONNECT^[91]

Disconnects the current dial-up connection.

EXTRACT^[91] *POS=[R]n TYPE=(TXT|HREF|TITLE|ALT) ATTR=Anchor**

Extracts data from websites.

FILEDELETE^[93] *NAME=file_name*

Deletes a file.

FILTER^[94] *TYPE=IMAGES STATUS=(ON|OFF)*

Filters web site elements.

Currently the support for filtering is experimental. If you need any other data filtered, please [let us know](#) what kind of filter you would like to see added.

FRAME^[94] *F=n*

Directs all following TAG^[110] or EXTRACT^[91] commands to the specified frame.

IMAGECLICK^[95] *IMAGE=image_file CONFIDENCE=n [CONTENT=some_content]*
Sends a WINCLICK^[42] command to the specified image.

IMAGESEARCH^[96] *IMAGE=image_file CONFIDENCE=n*
Searches for the the input image specified via the **IMAGE** attribute.

ONCERTIFICATEDIALOG^[97] *C=n*
Selects the client side certificate from a dialog.

ONDIALOG^[97] *POS=n BUTTON=(YES|NO|CANCEL) [CONTENT=some_content]*
Handles JavaScript dialogs.

ONDOWNLOAD^[98] *FOLDER=folder_name FILE=file_name*
Handles download dialogs.

ONERRORDIALOG^[99] *BUTTON=(YES|NO) CONTINUE=(YES|NO)*
Handles error dialogs.

ONLOGIN^[99] *USER=username PASSWORD=password*
Handles login dialogs.

ONSECURITYDIALOG^[101] *BUTTON=(YES|NO) CONTINUE=(YES|NO)*
Handles security dialogs.

ONWEBPAGEDIALOG^[101] *KEYS=some_keys*
Handles web page dialogs.

PRINT^[103]
Prints the current browser window.

PROMPT^[103] *prompt_text variable_name [default_value]*
Displays a popup to ask for a value. This value is stored in the variable.

PROXY^[104] *ADDRESS=proxy_URL:port [BYPASS=page_name]*
Connects to a proxy server to run the current macro.

REDIAL^[105] *ISP*

Redials a connection.

REFRESH^[106]

Refreshes (Reloads) current browser window.

SAVEAS^[106] *TYPE=(CPL/MHT/HTM/TXT/EXTRACT/BMP) FOLDER=folder_name*

FILE=file_name

Saves information to a file.

SET^[107] *variable_name variable_value*

Assigns values to built-in variables.

SIZE^[108] *X=n Y=m*

Resizes the iMacros Browser Window.

STOPWATCH^[109] *ID=id*

Measures time between two STOPWATCH commands.

TAB^[110] *T=(n/OPEN/CLOSE/CLOSEALLOTHERS)*

Sets focus on the tab with number n.

TAG^[110] *POS=n TYPE=type [FORM=form] ATTR=attr [CONTENT=some_content]*

Selects a webpage element.

URL^[113] *GOTO=some_URL*

Navigates to a URL in the currently active tab.

VERSION^[114] *BUILD=4213805*

Specifies the version of iMacros that created this macro.

WAIT^[114] *SECONDS=(n/#DOWNLOADCOMPLETE#)*

Waits for a specific time.

WINCLICK^[115] *X=n Y=n [CONTENT=some_content]*

Sends standard mouse clicks within the browser window to the specified coordinates [PRO and SCRIPTING Editions only].

8.1 ADD

Add a value to a variable. You can also subtract values by adding a negative value to the variable. If the content of at least one of the variables is not numeric, the values are concatenated as strings instead. The ADD command supports the built-in variables !VAR1^[131], !VAR2^[131], !VAR3^[132] and !EXTRACT^[122].

Syntax

ADD *result_var* *added_value*

Parameters

result_var

One of the four built-in variables !VAR1, !VAR2, !VAR3 or !EXTRACT. After the ADD command, this variable will hold the result.

added_value

The value that will be added to *result_var*

Examples

Both values are numeric, therefore the result is numeric as well:

```
SET !VAR1 100
ADD !VAR1 -20
=> Content of !VAR1 is 80.
```

At least one of the values is a string, therefore the result is a string as well:

```
SET !VAR1 Hello
ADD !VAR1 <SP>World<SP>
ADD !VAR1 2010
=> Content of !VAR1 is Hello World 2010
```

See Also

SET^[107]

8.2 BACK

Opens the previously visited website. Has the same effect as clicking the Back-button in your browser.

Syntax

BACK

Parameters

None

Examples

See Also

[CLEAR](#)^[88], [REFRESH](#)^[106]

8.3 CLEAR

Clears the browsers cache and all cookies on the hard drive. Can be useful, for example, to delete Web site cookies so every macro run starts at the some point. It is also useful to use this command before doing website response measurements.

Note:

Only data from the hard drive is deleted. Sometimes cookies are stored in the memory of the browser, those are not deleted. They are only deleted by restarting (closing and re-opening) of the browser window.

Syntax

CLEAR

Parameters

None

Examples

Use CLEAR command so that times of website response measurements are not influenced by using local cache data:

```
CLEAR
SET !FILESTOPWATCH mydata.csv
STOPWATCH ID=total
URL GOTO=http://www.iopus.com
```

STOPWATCH ID=total

See Also

[BACK](#)^[88], [STOPWATCH](#)^[109]

8.4 CLICK

"Clicks" on the element at the specified X/Y coordinates. The zero position (origin) of the coordinate system is in the top left corner of the webpage. The x axis is positive to the right, the y axis is positive downwards. The optional `CONTENT` attribute can be used to send text to the clicked element (e.g. if the clicked element is an input box).

In contrast to [WINCLICK](#)^[115], `CLICK` can click on elements of a page although they are not currently displayed in the browser window, i.e. this command does not need scrolling. Also in contrast to [WINCLICK](#)^[115], this click can be run in background mode.

Syntax

`CLICK X=n Y=m [CONTENT=some_content]`

Parameters

X

Specifies the x (horizontal) coordinate of the website element that is to be clicked.

Y

Specifies the y (vertical) coordinate of the website element that is to be clicked.

CONTENT

Specifies the value that is send to the clicked element.

Examples

The following example illustrates that the `CLICK` command can, as opposed to the [WINCLICK](#)^[115] command, "click" website elements that are not currently displayed in the browser window:

```
' open a iOpus homepage
URL GOTO=http://www.iopus.com/imacros/demo/v4/fl/form.asp
' reset size so that the big text box and the submit button are "hidden"
SIZE X=823 Y=300
' enter Remarks
CLICK X=458 Y=510 CONTENT=Without<SP>apples
' press submit
CLICK X=358 Y=644 CONTENT=Click<SP>to<SP>order<SP>now
```

See Also

[TAG](#)^[110], [WINCLICK](#)^[115]

8.5 CMDLINE

Sets a variable to a value retrieved from the command line. The most common use for this command is to read user input from the command line at runtime. It can also be used to assign a default value to a variable in case no value is set by the command line.

Syntax

`CMDLINE variable_name default_value`

Parameters

variable_name

Name of the variable that is to be set. Must be a variable that is changable via the command line, i.e. one of the following:

!TIMEOUT^[130]
!LOOP^[126]
!DATASOURCE^[119]
!VAR1^[131]
!VAR2^[131]
!VAR3^[132]
any dynamically defined variable^[25]

default_value

Default value *variable* is assigned if no value is given

Examples

Consider the following macro called macro1.iim:

```
CMDLINE !VAR1 abc.csv  
PROMPT Enter<SP>file<SP>name !VAR2 {{!VAR1}}
```

Starting this macro from the command line with (PRO Version and Scripting Edition only):

```
imacros.exe -macro macro1 -var1 efg.csv
```

would result in !VAR2 having the value efg.csv, while

```
imacros.exe -macro macro1
```

would result in !VAR2 having the value abc.csv.

See Also

SET^[107], !VARDEFAULT^[132], PROMPT^[103]

8.6 comments

' indicates a comment. If a line starts with ' everything behind the ' is ignored. Typically this is used for comments or to disable specific parts of a macro.

Note: **The macro cannot have empty lines**, as an empty line indicates the end of the macro. So every line in the macro must have at least the comment symbol.

8.7 DISCONNECT

Disconnects the current dial-up connection. On Dial-Up connections `DISCONNECT` and `REDIAL` can be used to change your IP address.

Syntax

`DISCONNECT`

Parameters

None

Examples

In order to get a new IP, use the following for a macro (change myISP in order to work with your connection):

```
'with old ip
URL GOTO=http://www.myipresolve.com/
DISCONNECT
REDIAL myISP
'with new ip
URL GOTO=http://www.myipresolve.com/
```

See Also

[REDIAL](#)^[105], [PROXY](#)^[104]

8.8 EXTRACT

Extracts data from websites. Use the attribute `ATTR` to determine which part is to be extracted. Normally, this part is generated by the Extraction Wizard. The `EXTRACT` command searches the HTML source code of the website for the `n`th occurrence of `ATTR` and extracts everything between the open (`<>`) and close (`</>`) tag that is last in `Anchor`. `Anchor` must always end with a wildcard `*`.

If in one macro several `EXTRACT` commands appear, the results are separated by the string `[EXTRACT]`. This tag is automatically translated into a line break when using the [SAVEAS](#)^[106] `TYPE=EXTRACT` command.

If [complete tables](#)^[37] where extracted, adjacent table elements are separated by the string #NEXT# and ends of table rows are delimited by the string #NEWLINE#. These tags are automatically translated into comas and newlines when you use the [SAVEAS](#)^[106] TYPE=EXTRACT command.

Extract Hidden Input Fields

To do this, record an EXTRACT command for a visible field (e.g. the name input field) and you get

```
EXTRACT POS=1 TYPE=TEXT ATTR=<INPUT*
```

we add the field name to the extraction anchor:

```
EXTRACT POS=1 TYPE=TEXT ATTR=<INPUT*abc*
```

Syntax

```
EXTRACT POS=[R]n TYPE=(TEXT|HREF|TITLE|ALT) ATTR=Anchor*
```

Parameters

POS

The number of the occurrence of the extraction anchor on the website. If this attribute is of the form Rn, the nth occurrence after a previously selected website element is extracted (Relative Extraction).

TYPE

Type of extraction.

TEXT

Plain text extraction, all HTML tags are taken out.

HREF

The URL of the page element the extraction anchor points to.

TITLE

The title of the page element the extraction anchor points to.

ALT

The alternative text of an image the extraction anchor points to.

ATTR

The extraction anchor. This attribute decides which part of the website is extracted. The wildcard * can be used. The extraction anchor *a*lways ends with an *.

Examples

Suppose the following HTML code is given and you would like to extract the text of the link in the second row, second column: bar

| | |
|----|---------------------|
| 1. | foo |
| 2. | bar |

```
<table>
  <tr>
    <td><a href="1.html">1.</a></td>
    <td><a href="foo.html">foo</a></td>
  </tr>
```

```
|  |  |
| --- | --- |
| <a href="1.html">2.</a></td>  <a href="bar.html">bar</a></td> | |

```

This can be done by extracting the text between the fourth TD tag:

```
EXTRACT POS=4 TYPE=TEXT ATTR=<TD>*
```

Or by extracting the text of the link tag:

```
EXTRACT POS=1 TYPE=TEXT ATTR=<A<SP>HREF="bar.html">*
```

Or by extracting the text of the next TD tag following the TD tag with the text 2. (relative extraction):

```

TAG POS=1 TYPE=TD ATTR=TEXT:2.
EXTRACT POS=R1 TYPE=TEXT ATTR=<TD>*

```

Or by extracting the fourth link using the wildcard *:

```
EXTRACT POS=4 TYPE=TEXT ATTR=<A<SP>HREF=*>*
```

Tips and tricks can also be found [here](#)^[30].

See Also

[TAG](#)^[110], [SAVEAS](#)^[106]

8.9 FILEDELETE

Deletes the file specified by Name. If no directory is specified in Name the file is assumed to lie in the "Downloads" subdirectory of the iMacros installation (e.g. C:\Program Files\iMacros\Downloads). You can also provide absolute paths to the file (e.g. C:\myData\lastDownloads.csv). Relative paths (e.g. ..\Macros\doExtract.iim) are not supported.

This command can be used to delete an old extraction file to avoid appending extraction results.

Syntax

```
FILEDELETE Name=file_name
```

Parameters

Name

Name of the file to be deleted.

Examples

Consider the case that a macro is extracting something from a webpage and saves it to the same file.

Usually, new data is appended to existing files. If you want the file to contain only the latest data, delete the file before saving.

See Also

8.10 FILTER

Filtering is new feature that allows you to change data on the website before it reaches the browser. Currently only the `TYPE=IMAGES` filter is supported. If enabled, it removes all references to images from the html source and thus speeds up page loading. *The filter command works for all macros until the filter is switched off or the iMacros Browser is closed.*

Currently the support for filtering is experimental. If you need any other data filtered, please [let us know](#) what kind of filter you would like to see added.

Syntax

```
FILTER TYPE=IMAGES STATUS=(ON|OFF)
```

Parameters

TYPE

Specifies the type of elements to filter. Currently, only `IMAGES` is supported

STATUS

Specifies whether the filter is turned on or off.

Examples

To start the filter and remove images, add this line to your macro:

```
FILTER TYPE=IMAGES STATUS=ON
```

To stop the filter:

```
FILTER TYPE=IMAGES STATUS=OFF
```

See Also

8.11 FRAME

Directs all following `TAG`^[110] or `EXTRACT`^[122] commands to the specified frame. The frame tag and number is automatically generated by clicking into a framed web page. The number of a frame is given by the position of its `<FRAME>` HTML tag in the `<FRAMESET>` context of the parent page.

If `EXTRACT` commands do not work on a framed website, this might be due to the fact that the focus is not in the frame in which the extraction should take place. In this case, try to click next to the information you want to extract before you start the extraction wizard.

Sometimes iMacros continues with a `TAG` command inside a frame before the web site in that frame has been completely loaded. This can be avoided by adding a `WAIT SECONDS=#DOWNLOADCOMPLETE#` command before the `TAG` command (see Examples).

Syntax

FRAME *F*=*n*

Parameters

F

Number of the frame.

Examples

Example macro: "Demo-Frames.iim"

```
FRAME F=6
TAG POS=1 TYPE=SPAN ATTR=TXT:Select<SP>path
```

This *could* generate an error if the content inside the frame is not loaded in time. To avoid this error, use

```
FRAME F=6
WAIT SECONDS=#DOWNLOADCOMPLETE#
TAG POS=1 TYPE=SPAN ATTR=TXT:Select<SP>path
```

instead.

See Also

`TAG`, `EXTRACT`, `TAB`

8.12 IMAGECLICK

Sends a `WINCLICK` command to the position of the input image specified via the `IMAGE` attribute. The input image is searched on the currently displayed website using image recognition algorithms. `CONFIDENCE` specifies how close the found image must match the input image. A value of 100 means exact match. This command is only available if the `Image Recognition Plugin` is installed. If the image is found, `!IMAGEX` and `!IMAGEY` hold the coordinates of where it was found.

Syntax

IMAGECLICK *IMAGE=image_file* *CONFIDENCE=n* [*CONTENT=some_content*]

Parameters

IMAGE

The path to local location of the image to be searched.

CONFIDENCE

The confidence level, i.e. level of similarity between the input and the remote picture. Must between 1 and 100.

CONTENT

The value that is sent to the clicked element.

Examples

Example macro: Demo-ImageRecognition

See Also

[WINCLICK](#)^[115], [IMAGESEARCH](#)^[96]

8.13 IMAGESEARCH

Searches for the the input image specified via the `IMAGE` attribute. The input image is searched on the currently displayed website using image recognition algorithms. `CONFIDENCE` specifies how close the found image must match the input image. A value of 100 means exact match. This command is only available if the [Image Recognition Plugin](#)^[60] is installed. If the input image is not found at the given confidence level, an error occurs.

If the image is found, [!IMAGEX](#)^[125] and [!IMAGEY](#)^[125] hold the coordinates of where it was found.

Syntax

```
IMAGESEARCH IMAGE=image_file CONFIDENCE=n
```

Parameters

IMAGE

The path to the image to be searched.

CONFIDENCE

The confidence level, i.e. level of similarity between the input and the remote picture. Must between 1 and 100.

Examples

Example macro: Demo-ImageRecognition

See Also

[WINLCLICK](#)^[115], [IMAGECLICK](#)^[95]

8.14 ONCERTIFICATEDIALOG

Selects the client side certificate on position *C* from the upcoming dialog.

Syntax

ONCERTIFICATEDIALOG *C=n*

Parameters

C

The position of the certificate in the list .

Examples

See Also

[ONDIALOG](#)^[97], [ONDOWNLOAD](#)^[98], [ONERRORDIALOG](#)^[99], [ONLOGIN](#)^[99], [ONSECURITYDIALOG](#)^[101], [ONWEBPAGEDIALOG](#)^[101]

8.15 ONDIALOG

Handles upcoming JavaScript dialogs. You can extract the text of a dialog by adding SET !EXTRACTDIALOG YES to your macro.

Tip: On some pages, a new page loads once a button on the dialog is clicked. If you want iMacros to wait for this page to load before continuing, please add WAIT SECONDS=#DOWNLOADCOMPLETE# after the TAG statement that triggers the dialog box.

Syntax

ONDIALOG *POS=n* *BUTTON=(YES|NO|CANCEL)* [*CONTENT=some_content*]

Parameters

POS

The position of the dialog. On some web pages, a few dialogs occur directly after each other. In this case, you can specify a specific decision for each dialog (*POS=1* is first dialog, *POS=2* is second dialog.) The *POS* statement applies only to dialogs that show directly after each other. This means that several ONDIALOG command with different *POS* values must occur before another macro command is executed.

BUTTON

Specifies which of the available buttons is pressed.

CONTENT

The content attribute is used to fill out input forms prompted by JavaScript.

Examples

Example macro: Demo-OnJavascriptDialog

See Also

[ONCERTIFICATEDIALOG](#)^[97], [ONDOWNLOAD](#)^[98], [ONERRORDIALOG](#)^[99], [ONLOGIN](#)^[99],
[ONSECURITYDIALOG](#)^[101], [ONWEBPAGEDIALOG](#)^[101]

8.16 ONDOWNLOAD

iMacros automatically detects and intercepts downloads. With this command, which has to occur before the download starts, the location and name of the saved file is determined.

The general web page timeout also applies to downloads. Make sure that the timeout value is long enough to cover the complete download time. Alternatively, you can add a [WAIT](#)^[114]

`SECONDS=#DOWNLOADCOMPLETE#`, which will cause iMacros to wait with the next command until the download has finished.

Normally the download file name is created automatically by the website. You can add parts to the original file name by using the + syntax (see Examples). The built-in variable [!NOW](#)^[127] is helpful in this context.

Syntax

```
ONDOWNLOAD FOLDER=folder_name FILE=file_name
```

Parameters

FOLDER

Specifies the folder in which the file is saved. Use * for the standard download folder specified in the options.

FILE

Specifies the file name under which the file is saved. If no file extension is given, the default file extension is used. Use * for the original file name. Use +something to add something to the original file name before the file extension.

Examples

Suppose, you download a file originally called `setup.exe`. The following snippet will save this file under the original file name in the default folder:

```
ONDOWNLOAD FOLDER=* FILE=*
```

This command will save the file in the `C:\MyFiles\` folder under the name `myDownload.exe`:

```
ONDOWNLOAD FOLDER=C:\MyFiles\ FILE=myDownload.exe
```

And finally, this command will save the file in the default directory under the name with an added date stamp (using the [!NOW](#)^[127] variable), e.g. `setup_060525.exe`.

```
ONDOWNLOAD FOLDER=* FILE=+{ { !NOW:yymmdd } }
```

Example macros: Demo-FileDownload, Demo-ImageDownload

See Also

[ONCERTIFICATEDIALOG](#)^[97], [ONDIALOG](#)^[97], [ONERRORDIALOG](#)^[99], [ONLOGIN](#)^[99],
[ONSECURITYDIALOG](#)^[101], [ONWEBPAGEDIALOG](#)^[101]

8.17 ONERRORDIALOG

If a page script error occurs on a webpage, Internet Explorer opens an error dialog. This command handles such a dialog so your macros are not interrupted by script errors. By default, the settings are `BUTTON=YES` and `CONTINUE=YES`. These settings are active even without an `ONERRORDIALOG` in your macro.

Syntax

```
ONERRORDIALOG BUTTON=(YES|NO) CONTINUE=(YES|NO)
```

Parameters

BUTTON

Specifies which button is clicked.

CONTINUE

Specifies if the macro is continued if such an error dialog comes up.

Examples

See Also

[ONCERTIFICATEDIALOG](#)^[97], [ONDIALOG](#)^[97], [ONDOWNLOAD](#)^[98], [ONLOGIN](#)^[99],
[ONSECURITYDIALOG](#)^[101], [ONWEBPAGEDIALOG](#)^[101]

8.18 ONLOGIN

Handles login dialogs. The `ONLOGIN` command must appear before the macro command that navigates to the site that brings up the login dialog. The password is stored with the method you selected on the Security tab of the options dialog. More details about the different password storage options can be found [here](#)^[13].

Syntax

```
ONLOGIN USER=username PASSWORD=password
```

Parameters

USER

Specifies the user name with which to log in.

PASSWORD

Specifies the password with which to log in.

Examples

```
ONLOGIN USER=username PASSWORD=password  
URL GOTO=http://www.mysavesite.com
```

See Also

[ONCERTIFICATEDIALOG](#)^[97], [ONDIALOG](#)^[97], [ONDOWNLOAD](#)^[98], [ONERRORDIALOG](#)^[99],
[ONSECURITYDIALOG](#)^[101], [ONWEBPAGEDIALOG](#)^[101]

8.19 ONPRINT

Handles print dialogs. The ONPRINT command must appear before the PRINT command which triggers the printer dialog to come up.

Syntax

```
ONPRINT P=n
```

Parameters

P

Specifies the number of the printer to use in the drop down list. If you use only P= or P=* the most recently selected printer is used. Typically this is the default windows printer.

Examples

```
URL GOTO=http://www.iopus.com  
ONPRINT P=3  
PRINT
```

See Also

[PRINT](#)^[103]

8.20 ONSECURITYDIALOG

Command to handle security dialogs. If `Continue=No` is selected, the macro will stop if such a dialog appears.

By default, the settings are `BUTTON=YES` and `CONTINUE=YES`. These settings are active even without an `ONSECURITYDIALOG` command in your macro.

Syntax

```
ONSECURITYDIALOG BUTTON=(YES|NO) CONTINUE=(YES|NO)
```

Parameters

BUTTON

Specifies which of the two buttons will be pressed.

CONTINUE

Specifies if macro continues if dialog appears.

Examples

See Also

[ONCERTIFICATEDIALOG](#)^[97], [ONDIALOG](#)^[97], [ONDOWNLOAD](#)^[98], [ONERRORDIALOG](#)^[99], [ONLOGIN](#)^[99], [ONWEBPAGEDIALOG](#)^[101]

8.21 ONWEBPAGEDIALOG

Web page dialogs are similar to Javascript dialogs, except they display HTML content. Since web page dialogs can contain any number of buttons or boxes, you can automate them by sending a specific list of keyboard commands to them. Web page dialogs are not normal HTML browser windows. Therefore they do not open in a separate tab, but pop up in front of the current window. The `ONWEBPAGEDIALOG` command must appear before the (in most cases) `TAG` command that triggers the popup.

During replay, `ONWEBPAGEDIALOG KEYS={WAIT<SP>2}{CLOSE}` is active by default to close unwanted ad dialogs.

Syntax

```
ONWEBPAGEDIALOG KEYS=some_keys
```

Parameters

KEYS

The series of key strokes send to the dialog. Please use `<SP>` as escape character for whitespaces and `
` for line breaks. Except for the alphanumeric (a-z, A-Z, 0-9 and so on) keys, the following

special keys and commands are supported (the curly brackets are mandatory):

```
{TAB}
```

Enter a tabulator.

```
{ENTER}
```

Hit Enter key.

```
{CLOSE}
```

Close the dialog

```
{WAIT<SP>Seconds}
```

Wait for *Seconds* seconds before proceeding with the next key stroke.

Commands can be repeated by adding a number to it. For example `{tab<SP>3}` is the same as writing `{tab}{tab}{tab}`.

Examples

Related example macro: Demo-OnWebPageDialog

If you want to close all upcoming dialogs immediately, use

```
ONWEBPAGEDIALOG KEYS={CLOSE}
```

Suppose a dialog comes up that asks you to enter your name and country, use

```
ONWEBPAGEDIALOG KEYS=Frank<SP>Miller<BR>USA{ENTER}
```

See Also

[ONCERTIFICATEDIALOG](#)^[97], [ONDIALOG](#)^[97], [ONDOWNLOAD](#)^[98], [ONERRORDIALOG](#)^[99], [ONLOGIN](#)^[99], [ONSECURITYDIALOG](#)^[101]

8.22 PAUSE

Same as a manual click of the "Pause" button: Stops the execution of the macro. Waits for user to click "Continue" to continue.

The command is useful if the macro should wait for manual user input or for debugging.

Syntax

```
PAUSE
```

Parameters

None

Examples

Related example macro: -

See Also

[PROMPT](#)^[103]

8.23 PRINT

Prints the current browser window on your default printer.

If the page uses frames and you want to print only a specific frame, select this frame with [WINCLICK](#)^[42] first.

Tip: If you want to print PDF files with the Acrobat Reader Plug-In, you need to add [ONDIALOG](#)^[97] POS=1 BUTTON=OK command before the PRINT statement.

Syntax

PRINT

Parameters

None

Examples

Related example macro: Demo-Print

See Also

[ONPRINT](#)^[100], [ONDIALOG](#)^[97]

8.24 PROMPT

Displays a popup to ask for a value. This value is stored in *variable_name*. This command can only be used to change the variables [!VAR1](#)^[131], [!VAR2](#)^[131] or [!VAR3](#)^[132], but not built-in variables like [!DATASOURCE](#)^[119] or dynamically generated variables.

Syntax

PROMPT *prompt_text variable_name [default_value]*

Parameters

prompt_text

The text which is to be displayed above the input text field. Remember to use the escape character

<SP> for whitespaces.

variable_name

Name of the variable to save the user input in.

default_value

Default value which will be displayed in the input field when the dialog comes up. Must not be a value of a built-in variable except [!VAR1](#)^[131], [!VAR2](#)^[131] or [!VAR3](#)^[132].

Examples

If you do not want to save passwords (even encrypted) in on your computer, use the following code to ask for the password:

```
PROMPT Please<SP>enter<SP>your<SP>password: !VAR1
```

Suppose you would like to ask for more then three inputs from the user. Since you only have three variables to use the `PROMPT` command for, you might run into problems. The solution is to navigate to the site where you are going to fill the values in, then ask for the first three inputs, fill them into the website, ask for the next three values, fill them in etc.

See Also

[SET](#)^[107], [CMDLINE](#)^[90], [PAUSE](#)^[102]

8.25 PROXY

Connect to a proxy server to run the current macro. The iMacros Browser then connects to the Internet through a proxy server by using the settings you specify. You can define a specific proxy server for each macro. Each running instances of the iMacros Browser can have its own proxy server. If you have proxy server settings defined in the Internet Explorer settings, you will not need to use a `PROXY` command since iMacros automatically applies the Internet Explorer settings.

Syntax

```
PROXY ADDRESS=proxy_URL:port [BYPASS=page_name]
```

Parameters

ADDRESS

The URL and port of the proxy server. You can define separate proxy servers for http and https connections (see the example below).

BYPASS

The URLs for which the Proxy server is not to be used. If you want to connect to a computer on your intranet, make sure you include its address here (see the example below). If no `BYPASS` is specified, the default setting is used, which means no proxy server is used for local addresses inside your network. You may use the wildcard `*`.

Examples

The following command uses a local proxy server for both http and https at the address 192.1.8.1 and the port number 8080. Since no bypass is specified, the default settings are used.

```
PROXY ADDRESS=192.1.8.1:8080
```

This command specifies two different proxy server for the http and https protocol. Defines no bypass so iMacros uses these proxy servers even for local addresses.

```
PROXY ADDRESS=http=192.1.8.1:8080<SP>https=192.1.8.2:8080 BYPASS=NULL
```

To use a proxy server at address 66.98.229.110, but not to use it for URLs including the word "iopus", use

```
PROXY ADDRESS=66.98.229.110:8080 BYPASS=*iopus*
```

You can also use the same command, but with URL instead of IP address.

```
PROXY ADDRESS=www.iopus.com:8080 BYPASS=*iopus*
```

See Also

[REDIAL](#)^[105], [DISCONNECT](#)^[91]

8.26 REDIAL

Redials a connection.

Syntax

```
REDIAL ISP
```

Parameters

ISP

Specifies the name of your dialup connection as it appears in the Windows dialup selection.

Examples

In order to get a new IP, use the following for a macro (change myISP in order to work with your connection):

```
'with old ip
URL GOTO=http://www.myipresolve.com/
DISCONNECT
REDIAL myISP
'with new ip
URL GOTO=http://www.myipresolve.com/
```

See Also

[DISCONNECT](#)^[91], [PROXY](#)^[104]

8.27 REFRESH

Refreshes (Reloads) current browser window. Refresh includes sending a "pragma:nocache" header to the server (HTTP URLs only) which causes all elements of the website to be reloaded from the webserver.

Syntax

REFRESH

Parameters

None

Examples

Use the REFRESH and [STOPWATCH](#)^[109] commands to do a statistical measurement of response times:

```
SET !FILESTOPWATCH myresults.csv
STOPWATCH ID=total
URL GOTO=http://www.iopus.com
STOPWATCH ID=total
STOPWATCH ID=total
REFRESH
STOPWATCH ID=total
STOPWATCH ID=total
REFRESH
STOPWATCH ID=total
...
```

See Also

[BACK](#)^[88], [CLEAR](#)^[88]

8.28 SAVEAS

Saves information to a file. The SAVEAS command can save different information to a file. You can add parts to the original file name by using the + syntax (see Examples). The built-in variable [!NOW](#)^[127] is helpful in this context.

Syntax

SAVEAS TYPE=(CPL|MHT|HTM|TXT|EXTRACT|BMP) FOLDER=folder_name FILE=file_name

Parameters

TYPE

Specifies the type of the save information. The following options are available:

CPL

The complete web page is saved. The files and images are saved separately and stored in a folder.

MHT

The web page plus images are saved in a single file (Web Archive).

HTM

The web page source is saved with no images. If the page has frames, all framed HTML pages are saved automatically.

TXT

Only the web page text is saved; all HTML tags are omitted.

EXTRACT

The value of the variable !EXTRACT is saved in CSV format.

BMP

A [screenshot](#)^[14] of the web page is saved.

FOLDER

Specifies the folder in which the file is saved. Use * for the standard download folder specified in the Paths tab of the Options dialog, e.g. C:\Program Files\iMacros\downloads\.

FILE

Specifies the file name under which the file is saved. If no file extension is given, the default file extension is used. If you use FILE=*, the default file name extract.csv is used. Use +something to add something to the original file name before the file extension.

Examples

Save the current webpage in a file called homepage_current.mht

```
SAVEAS TYPE=MHT FOLDER=* FILE=homepage_current
```

Save a screenshot of the current page with the current date in the original filename

```
SAVEAS TYPE=BMP FOLDER=C:\Screenshots FILE=+{!NOW:ddmmyyyy}
```

See Also

[ONDOWNLOAD](#)^[98], [EXTRACT](#)^[91]

8.29 SET

This command assigns values to [built-in variables](#)^[116]. Dynamically generated variables cannot be set using this command. SET !LOOP n is ignored for every loop run except the first one.

Syntax

```
SET variable_name variable_value
```

Parameters

variable_name

Specifies the name of the variable which is to be set. Must be a [built-in variable](#)^[116].

variable_value

Specifies the value to which the variable is to be set.

Examples

A default value is assigned to [!VAR1](#)^[131] and then it is asked if that is correct using the [PROMPT](#)^[103] command.

```
SET !VAR1 Barney  
PROMPT What<SP>is<SP>your<SP>dogs<SP>name? !VAR1 {{ !VAR1 }}
```

See Also

[PROMPT](#)^[103], [CMDLINE](#)^[90]

8.30 SIZE

Resizes the iMacros Browser Window. This command does *not* work when iMacros is `-silent` or `-tray` mode. In Internet Explorer, this command is ignored.

Syntax

```
SIZE X=n Y=m
```

Parameters

X

Specifies the new width of the iMacros Browser window in pixel.

Y

Specifies the new height of the iMacros Browser window in pixel.

Examples

When using [WINCLICK](#)^[115], this size of the browser window is essential for the success of the command. If the window is resized between recording and replaying the macro, the [WINCLICK](#)^[115] command might fail, because the position of the element which is to be clicked has moved. Therefore, it is advisable (especially when you distribute your macros) to add a `SIZE` command before all [WINCLICK](#)^[115] commands, so that potential window resizing of the user is revoked.

```
SIZE X=800 Y=600  
WINCLICK X=234 Y=42 CONTENT=Hello<SP>World!
```

See Also

[WINCLICK](#)^[115], [IMAGECLICK](#)^[95]

8.31 STOPWATCH

Measures the time in seconds between to STOPWATCH command with the same identifier.

This command can be used for website [response time measurement](#)^[44]. Using different IDs in one macro, different processes can be timed separately from each other. By default, the measured data is stored in the download directory of iMacros. You can change the file location with the [!FILESTOPWATCH](#)^[124] variable. (PRO and SCRIPTING Editions only).

Syntax

STOPWATCH ID=*id*

Parameters

ID

Specifies an unique identifier for a timing measurement. Up to 100 different identifiers can be used.

Examples

```
VERSION BUILD=4230323
SET !FILESTOPWATCH mydata.csv
STOPWATCH ID=total
URL GOTO=http://www.iopus.com/iim/
STOPWATCH ID=1
TAG POS=1 TYPE=A ATTR=HREF:http://www.iopus.com/iim/compare
STOPWATCH ID=1
STOPWATCH ID=store
TAG POS=1 TYPE=A ATTR=TEXT:US$<SP>149
TAG POS=1 TYPE=INPUT:SUBMIT FORM=NAME:order
ATTR=NAME:ORDER_PRODUCT_NOW&&VALUE:Order<SP>Now
TAG POS=1 TYPE=A ATTR=HREF:http://www.iopus.com/store
STOPWATCH ID=store
STOPWATCH ID=total
```

The file "mydata.csv" contains the measurement data. By default, the data is saved to the Downloads\ directory of your iMacros installation. The values are comma separated (CSV format) so they can be viewed with any text editor or imported directly in Excel or any other software you use to view your data.

Example file

Format: Date, Time, Macro name, ID, time (s)

```
"2004/08/3","11:56:23","mymacro","1","1.272"
"2004/08/3","11:56:32","mymacro","store","8.943"
"2004/08/3","11:56:32","mymacro","total","10.21"
```

See Also

[REFRESH](#)^[106], [CLEAR](#)^[88]

8.32 TAB

Sets focus on the tab with number *n*.

Syntax

`TAB T=(n | OPEN | CLOSE | CLOSEALLOTHER)`

Parameters

T

Specifies the tab number. Other than a number, the following commands are possible:

CLOSE

Closes the current tab.

OPEN

Opens a new tab in the background.

CLOSEALLOTHER

Closes all tab except the one currently in focus.

Examples

Related example macros: Demo-Tab

See [examples](#)^[22].

If you want to open a second website, use the following code:

```
' open a webpage in the first tab
URL GOTO=http://www.iopus.com
' open a new tab
TAB OPEN
' get new tab to foreground
TAB T=2
' load another page
URL GOTO=http://www.google.com
' close the second tab
TAB CLOSE
TAB T=1
```

See Also

[FRAME](#)^[94], [TAG](#)^[110]

8.33 TAG

The TAG command selects HTML elements from the current website. The identification of the element is given by providing the parameters POS, TYPE, FORM and ATTR. If the selected element is

a link, the link is followed, i.e. the TAG command acts as if it clicks on the element. If the selected element is part of an input form, the CONTENT parameter can be used to fill content into the selected field.

For the TYPE, FORM and ATTR parameter sub-parameters are used. Sub-parameters are used in the form

parameter=sub-parameter:value.

Especially in the ATTR parameter extensive use of the wildcard * can be made.

Following links

To follow links, the TYPE parameter of the TAG command is set to A. For text links the FORM parameter is not needed. Which link will be followed is solely determined by the POS and the ATTR parameter. Except for the number of occurrence on the page (determined by the POS parameter), a link is uniquely identified by its name and its URL. Therefore, the ATTR parameter offers two different sub-parameters: TXT and HREF. To select a link by its name, use

ATTR=TXT:some_name,

to select by its URL, use

ATTR=HREF:some_url.

Sometimes iMacros continues with a TAG command inside a frame before the web site in that frame has been completely loaded. This can be avoided by adding a [WAIT](#)^[114]

SECONDS=#DOWNLOADCOMPLETE# command before the TAG command.

Filling forms

On one webpage several forms can appear, therefore the FORM parameter determines which form to use by its NAME sub-parameter. The input type is selected using the TYPE parameter, if necessary with the sub-parameter INPUT.

Special TAG Commands for Drop Down Menus

The CONTENT parameter is used to specify which element of a drop down menu (generated by the HTML SELECT command) is selected. By default, the CONTENT parameter of TYPE=SELECT stores the position in the list (also called index) of the selected value. This typically looks like this

TAG TYPE=SELECT FORM=NAME:form1 ATTR=NAME:select1 CONTENT=1

You can manually change this so that an entry is selected by its name or its value. The name of the menu item is the text which is displayed in the drop down menu. The value is the text that follows the VALUE= parameter in the OPTION tag. To select an entry by its name, add a \$ before the name so that the command looks like

TAG TYPE=SELECT FORM=NAME:form1 ATTR=NAME:select1 CONTENT=\$Apple

To select an entry by its value, use the percent symbol (%)

TAG TYPE=SELECT FORM=NAME:form1 ATTR=NAME:select1 CONTENT=%AP

Spaces must be written as <SP>. The comparison in the latter two cases is case insensitive and wildcards can be used.

In multiple selection menus, multiple selection can be achieved by listing indices separated by a colon (:). To select all values of a selection, use CONTENT=ALL.

If you start the macro via iimPlay and select a value that is NOT in the selection, the iimPlay command returns an error code that is the maximum number of lines possible.

Hidden Input

Hidden input fields are used in HTML to write information (like usernames or session IDs) in the HTML code so that the user does not have to type them in manually every time a new page is loaded. Since hidden fields are not displayed in the browser, the TAG command for them cannot be generated during normal recording. But you can add TAG commands with TYPE=HIDDEN manually into the macro and iMacros will fill these hidden fields during replay. To do this, record an TAG command for a visible field (e.g. the name input field) and you get:

TAG POS=1 TYPE=INPUT:TEXT FORM=NAME:TestForm ATTR=NAME:Name CONTENT=Peter

To modify this for the hidden field we change the `TYPE` from `INPUT:TEXT` to `INPUT:HIDDEN` and the name from `Name` to `abc`:

```
TAG POS=1 TYPE=INPUT:HIDDEN FORM=NAME:TestForm ATTR=NAME:abc CONTENT=999999
```

File Upload

Internet Explorer and other browsers do not allow automatic filling of any fields with the type `FILE`. Typically these fields are used for uploading files. However, iMacros can work around this limitation with the `WINCLICK` statement. More details can be found in the "Demo-FileUpload" macro.

Triggering events

Sometimes it is required to fire certain events concerning a web site element characterized by the parameters in the TAG command. To fire an event, set `EVENT:#event_name` as the `CONTENT` parameter. Currently, two events are supported, `SAVEITEM` and `MOUSEOVER`.

SAVEITEM

This event can be used to save items, mostly images, from any web site. A command doing this is most easily compiled by using the [Save Item Wizard](#)^[16] or clicking on the elements during record and adding `CONTENT=EVENT:#SAVEITEM` manually. To control the location and name of the downloaded file, use the [ONDOWNLOAD](#)^[98] command.

SAVEPICTUREAS

The command is named after the corresponding functions in the Internet Explorer menu, "Save Picture as". You can right-click on any web page element to see if the web page element supports this feature. To create such a command, you have to click on the elements during record and add `CONTENT=EVENT:#SAVEITEM` manually. To control the location and name of the downloaded file, use the [ONDOWNLOAD](#)^[98] command.

SAVETRARGET

The command is named after the corresponding functions in the Internet Explorer menu, "Save Target as". You can right-click on any web page element to see if the web page element supports this feature. To create such a command, you have to click on the elements during record and add `CONTENT=EVENT:#SAVEITEM` manually. To control the location and name of the downloaded file, use the [ONDOWNLOAD](#)^[98] command.

MOUSEOVER

For the selected element a mouseover event is triggered. This is sometimes needed to change the state of an image.

FAIL_IF_FOUND

This event is an iMacros internal event as oppose to an HTML event. It is useful when doing [keyword searches](#)^[79].

Syntax

```
TAG POS=n TYPE=type [FORM=form] ATTR=attr [CONTENT=some_content]
```

Parameters

POS

Specifies the number of the occurrence of the HTML element. By default, the occurrence is counted from the top of the page. But using "POS=Rx" the occurrence can also be relative to the previous TAG command (see [Relative Extraction](#)^[34] for more details on this concept.)

TYPE

Specifies the HTML type of the element

FORM

Specifies the name of the parent FORM element (only valid for FORM child elements).

ATTR

Specifies the identifier for the selected element.

CONTENT

Specifies the content to send to the selected element (only valid for FORM child elements).

Examples

See any of the example demos in the `Macros\` directory of your iMacros installation.

See Also

[FRAME](#)^[94], [TAB](#)^[110], [EXTRACT](#)^[91], [CLICK](#)^[89], [WINCLICK](#)^[115], [BACK](#)^[88], [URL](#)^[113]

8.34 URL

Navigates to a URL in the currently active tab.

Syntax

`URL GOTO=some_URL`

Parameters

GOTO

Specifies the URL to go to.

Examples

If you want to open a second website, use the following code:

```
' open a webpage in the first tab
URL GOTO=http://www.iopus.com
' open a new tab
TAB OPEN
' get new tab to foreground
TAB T=2
' load another page
URL GOTO=http://www.google.com
```

See Also

[TAB](#)^[110], [TAG](#)^[110], [BACK](#)^[88]

8.35 VERSION

Specifies the version of iMacros that created this macro.

Syntax

```
VERSION BUILD=build
```

Parameters

BUILD

Specifies the build number

Examples

```
VERSION BUILD=421805
```

corresponds to Version 4.21 (Build 805).

See Also

8.36 WAIT

Waits for a specific time before continuing replay with the next command.

Sometimes iMacros continues with a [TAG](#)^[110] command inside a frame before the web site in that frame has been completely loaded. This can be avoided by adding a `WAIT SECONDS=#DOWNLOADCOMPLETE#` command before the [TAG](#)^[110] command (see Examples).

Syntax

```
WAIT SECONDS=( n / #DOWNLOADCOMPLETE# )
```

Parameters

SECONDS

Specifies the number of seconds to wait. If a download was started, use `#DOWNLOADCOMPLETE#` to wait until the download has finished.

Examples

Follow a link on a framed site:

```
FRAME F=6
```

```
TAG POS=1 TYPE=SPAN ATTR=TXT:Select<SP>path
```

This *could* generate an error if the content inside the frame is not loaded in time. To avoid this error, use

```
FRAME F=6
WAIT SECONDS=#DOWNLOADCOMPLETE#
TAG POS=1 TYPE=SPAN ATTR=TXT:Select<SP>path
```

instead.

See Also

[ONDOWNLOAD](#) 

8.37 WINCLICK

The WINCLICK sends standard mouse clicks within the browser window to the coordinates specified by the *X* and *Y* parameter (PRO and SCRIPTING Editions only). Typically, WINCLICK is used to automate web pages that contain non-HTML elements such as Java-applets or Macromedia Flash elements. The WINCLICK command can send keystrokes to the web browser via the *CONTENT* attribute.

The difference between the WINCLICK and the CLICK command

- The WINCLICK commands operates on the visible part of the web page, just a like human would with a mouse. Thus WINCLICK *X*=1 *Y*=400 will always click on the specific part of the browser window. If the web page is scrolled, the click hits another part of the web page.
- The CLICK command operates on the complete browser web page (HTML) only. So CLICK *X*=1 *Y*=980 will always click on the specific HTML element at this position of the page regardless of whether the element is currently visible in the browser window or not. Whether or not the web page is scrolled, a specific *X/Y* combination always hits the same part of the web page.

Syntax

```
WINCLICK X=n Y=m [CONTENT=some_content]
```

Parameters

X

Specifies the *x* (horizontal) coordinate of the website element that is to be clicked.

Y

Specifies the *y* (vertical) coordinate of the website element that is to be clicked.

CONTENT

Specifies the value that is send to the clicked element. In addition to regular text, it can send special keys:

```
{ENTER}
Send an Enter (Return) keystroke.
{TAB}
Send a Tabulator keystroke.
{DEL}
```

Send a Delete keystroke.

{BACKSPACE}

Send a Backspace keystroke.

{LEFT}

Send a Left Cursor keystroke.

{RIGHT}

Send a Right Cursor keystroke.

Examples

Related example macros: Demo-Winclick, Demo-FileUpload, Demo-Flash, Demo-JavaScriptMenu

See Also

[TAG](#)^[110], [CLICK](#)^[89]

9 Built-in Variables

Available built-in variables:

All built-in variables start with an exclamation mark and are all caps. Variables that you create via the [command line](#)^[50] or the [Scripting Interface](#)^[53] must not include an exclamation mark and are not case sensitive, which means, for example, that {{MyID}} is the same as {{myid}}. All built-in variables are changed manually using the [SET](#)^[107] command.

[!COLn](#)^[119]

Specifies the column which is used for input.

[!DATASOURCE](#)^[119]

Specifies an input file for merging data with macro.

[!DATASOURCE_COLUMNS](#)^[120]

Specifies the number of columns in the input [datasource](#)^[119].

[!DATASOURCE_LINE](#)^[120]

Specifies the line in the [datasource](#)^[119] which is used for input.

[!DIALOGMANAGER](#)^[120]

Activates or stops the [Dialog Manager](#)^[18].

[!ENCRYPTION](#)^[121]

Specifies how to encrypt passwords you use in macros.

!ERRORIGNORE^[121]

Tells iMacros to ignore errors.

!ERRORMACRO^[122]

Overrides the global error macro setting in the Options dialog for the current macro.

!EXTRACT^[122]

Contains the extraction results.

!EXTRACT TEST POPUP^[123]

Toggles whether the extraction result is displayed during replay.

!EXTRACTADD^[123]

Adds (appends) a value to the extraction results.

!EXTRACTDIALOG^[123]

Extract information from a dialog^[18].

!FILELOG^[124]

Sets a specific log file name for the current macro.

!FILESTOPWATCH^[124]

Sets the file name for the file that contains the stopwatch measurement^[44] data.

!FOLDERIMACROS^[124]

Returns the folder from which the "imacros.exe" file was started ("Application Path").

!IMAGEX^[125]

This value contains the X-coordinate of the last image found with the IMAGESEARCH^[96] or IMAGECLICK^[95] command.

!IMAGEY^[125]

This value contains the Y-coordinate of the last image found with the IMAGESEARCH^[96] or IMAGECLICK^[95] command.

!LOADCHECK^[125]

Switches the LoadCheck on or off.

!LOOP^[126]

Counts the current loop number in loop mode.

!MACROTIMEOUT^[126]

Specifies the maximal runtime for the entire macro.

!NOW^[127]

Contains the current time and date.

!POINTER^[129]

Disables or enables the pointer (blue frame) that marks the current position during a macro replay.

!REPLAYSPEED^[129]

Sets the replay speed.

!STOPWATCHTIME^[130]

Contains the last measured response time^[45] value.

!TIMEOUT^[130]

Sets the timeout value in seconds.

!URLCURRENT^[130]

Contains the current URL.

!URLSTART^[131]

Contains the URL that was active in the browser when the macro started.

!VAR1^[131]

One of the three standard built-in variables for arbitrary use.

!VAR2^[131]

One of the three standard built-in variables for arbitrary use.

!VAR3^[132]

One of the three standard built-in variables for arbitrary use.

[!VARDEFAULT](#)^[132]

Holds a value which is assign automatically to all undefined variables.

9.1 !COLn

Specifies the column which is used for input. Set n to the column number you want to use.

Value

Any positive integer greater than 0 and less the [!DATASOURCE_COLUMNS](#)^[120].

Examples

In this example, the example.csv file holds the first names in the first and the last names in the second column. On a web site, we want to insert these values into the appropriate fields.

```
SET !DATASOURCE example.csv
SET !DATASOURCE_COLUMNS 2
SET !DATASOURCE_LINE {{!LOOP}}
URL GOTO=http://www.some_input.com/enter_name.html
TAG POS=1 TYPE=INPUT:TEXT FORM=form1 ATTR=NAME:first_name CONTENT={{!COL1}}
TAG POS=1 TYPE=INPUT:TEXT FORM=form1 ATTR=NAME:last_name CONTENT={{!COL2}}
```

More examples [here](#)^[27].

See Also

9.2 !DATASOURCE

Specifies the name and location of an input file for merging data with macro. If no folder is supplied, the file is assumed to lie in the standard data input directory of your iMacros installation, e.g.

C:\Program Files\iMacros\datasources\.

Value

Valid name and location of the input file.

Examples

```
SET !DATASOURCE C:\mysource.txt
```

More examples [here](#)^[27].

See Also

9.3 !DATASOURCE_COLUMNS

Specifies the number of columns in the input [datasource](#)^[119].

Value

Valid name and location of the input file.

Examples

```
SET !DATASOURCE_COLUMNS 3
```

More examples [here](#)^[27].

See Also

9.4 !DATASOURCE_LINE

Specifies the line in the [datasource](#)^[119] which is used for input.

Value

Any positive integer.

Examples

```
SET !DATASOURCE_LINE { { !LOOP } }
```

More examples [here](#)^[27].

See Also

9.5 !DIALOGMANAGER

Activates or stops the [Dialog Manager](#)^[18].

Value

YES | NO

Examples

If you want your users to handle the print dialog manually, set this variable to NO before the [PRINT](#)^[103]

command.

```
SET !DIALOGMANAGER NO
PRINT
SET !DIALOGMANAGER YES
```

See Also

9.6 !ENCRYPTION

Specifies how to encrypt passwords you use in macros. Applies only to values of password fields. Password fields are the ones where words are displayed using asterisks (*). Setting this variable in your macro will overwrite the settings in the Options -> Security tab.

1. No encryption

The password is stored inside the macro in plain text. To activate this option, set !ENCRYPTION to NO.

2. Encrypted web site passwords

Passwords are encrypted by a master password that is stored on your own computer. To activate this option, set !ENCRYPTION to STOREDKEY.

3. Encrypted web site passwords and ask for the Master Password

The master password is not stored. It is only kept *temporarily* in memory while you run the macros. To activate this option, set !ENCRYPTION to TMPKEY.

More information [here](#)^[13].

Value

NO | STOREDKEY | TMPKEY

Examples

```
SET !ENCRYPTION TMPKEY
```

See Also

9.7 !ERRORIGNORE

Tells iMacros to ignore errors. The replay of macros continues even if one or more commands fail. If you set this parameter to YES, [LOADCHECK](#)^[125] is automatically disabled as well.

Value

YES | NO

Examples

See Also

[Error Handling](#)^[48]

9.8 !ERRORMACRO

Instructs iMacros to run a specific macro if an error occurs. This overrides the global error macro setting in the options dialog for the current macro.

Value

Valid macro name.

Examples

```
SET !ERRORMACRO myErrorMacro
```

See Also

[Error Handling](#)^[48]

9.9 !EXTRACT

Contains the extraction results. After the expression `{{ !EXTRACT }}` is used once, the content is reset. If you want to use it in more than one place, save it to another variable first using the [SET](#)^[107] command.

If in one macro several [EXTRACT](#)^[91] commands appear, the results are separated by the string `[EXTRACT]`. This tag is automatically translated into a line break when using the [SAVEAS](#)^[106] `TYPE=EXTRACT` command.

If [complete tables](#)^[37] were extracted, adjacent table elements are separated by the string `#NEXT#` and ends of table rows are delimited by the string `#NEWLINE#`. These tags are automatically translated into comas and newlines when you use the [SAVEAS](#)^[106] `TYPE=EXTRACT` command.

Value

Extraction result or `#EANF#` (Extraction Anchor Not Found).

Examples

```
TAG . . . . CONTENT={{ !EXTRACT }}
```

```
SET !EXTRACT NULL (Reset content of EXTRACT)
```

[See Also](#)

9.10 !EXTRACT_TEST_POPUP

Toggles whether the extraction result is displayed during replay in a popup dialog. Set this value to **NO** to disable the extract testing popup. The default value is **YES** unless the macro was started via the command line or the Scripting Interface.

Value

YES | NO

Examples

[See Also](#)

9.11 !EXTRACTADD

Adds (appends) a value to the extraction results. This new value is separated by [EXTRACT] tag from previous extraction results, just as normal extraction results. Adding a specific value or variable to the extraction results is often useful if you want to add an input parameter (for example a search keyword that you read from a CSV list) to your output file of extraction results.

Value

Any character or combination of characters.

Examples

Suppose you want to add the input to a website, which is stored in the variable !COL1, to be also stored in the output. The easiest way to accomplish this is

```
SET !EXTRACTADD {{!COL1}}
```

Instead of using !EXTRACTADD, you can also use

```
ADD !EXTRACT {{!COL1}}
```

[See Also](#)

9.12 !EXTRACTDIALOG

Extract information from a [dialog](#)^[18]. The entire text of a website dialog is extracted. The text is processed the same way as standard [EXTRACT](#)^[91] text.

Valid Value

YES | NO

Examples

See Also

9.13 !FILELOG

Sets a specific log file name for the current macro. If no folder is supplied, the file will be written to the standard `download` directory of your iMacros installation. If `!FILELOG` is not set, all information is logged to the main log file specified in the Options dialog.

Value

Valid name and location of the output file.

Examples

See Also

9.14 !FILESTOPWATCH

Sets the file name and location for the file that contains the [stopwatch measurement](#)^[44] data. By default the file name is `stopwatch.csv` and is located in the `download` directory of your iMacros installation. If no folder is supplied, the file will be written to the standard `download` directory of your iMacros installation. Setting `!FILESTOPWATCH` to `NO` instructs iMacros *not* to create a response time log file.

Value

Valid name and location of the output file|NO.

Examples

See Also

9.15 !FOLDERIMACROS

Returns the folder from which the `imacros.exe` file was started ("Application Path"). This path can be useful if you use iMacros with *local* web pages (Offline Mode) and you need a relative path.

Value

Application path of the iMacros instance.

Examples

URL GOTO=file:///{{!FOLDERIMACROS}}/local/test.htm

See Also

9.16 !IMAGEX

This value contains the X-coordinate of the last image found with the [IMAGESEARCH](#)^[96] or [IMAGECLICK](#)^[95] command.

Value

X-coordinate in pixel.

Examples

See Also

9.17 !IMAGEY

This value contains the Y-coordinate of the last image found with the [IMAGESEARCH](#)^[96] or [IMAGECLICK](#)^[95] command.

Value

Y-coordinate in pixel.

Examples

See Also

9.18 !LOADCHECK

Switches LoadCheck on (YES) or off (NO). The default value is on. LoadCheck runs if a TAG command fails, i.e. the associated HTML element is not found. Before an error occurs, iMacros does some additional checks to make sure the HTML element is really not found. Typically, this check is required only for complex pages with frames and JavaScript and pages that reload themselves. LoadCheck is associate with waiting a certain amount of time, which is automatically 1/10th of the [!TIMEOUT](#)^[130]

value. The `!TIMEOUT`^[130] default value is 60s, thus the LoadCheck default value is 6s. If you set `!ERRORIGNORE`^[121] to YES, LoadCheck is automatically switched off.

Value

YES | NO

Examples

Related example macros: Demo-FillVariousWebsites

See Also

9.19 !LOOP

Counts the current loop number in loop mode. Especially useful together with the POS attribute of the `TAG`^[110] command. With `SET !LOOP 3` you can set a start value for the loop counter (the default value is 1). `SET !LOOP n` is ignored for every loop run except the first one.

Value

Any integer in the Scripting Edition.
Any integer less than 1000 in the PRO Version.
Any integer less than 100 in the Power User Edition

Examples

Related example macros: Demo-Slideshow

See Also

9.20 !MACROTIMEOUT

Specifies the maximal runtime for the entire macro. The macro level timeout creates a timeout error (-321) if the overall macro run time exceeds this value.

A timeout value of 0 means no timeout. The default is no macro level timeout.

Value

Any positive integer or 0.

Examples

See Also

9.21 !NOW

Contains the current time and date. In order to format the time and date you can use the following format codes, which you need to append to the variable after a colon (see Examples):

`!NOW:format_code`

The format codes are case sensitive. Format code can include spaces but make sure to use the `<SP>` escape character. If you want to include text like `th` to format something like `June 4th`, enclose the text in `"`.

| | |
|--------|--|
| none | Displays the number with no formatting, i.e. the standard format defined by your operating system is used.. |
| : | Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings. This option should NOT be used in file names. |
| / | Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings. This option should NOT be used in file names. |
| D | Displays the day as a number without a leading zero (1 - 31) |
| Dd | Displays the day as a number with a leading zero (01 - 31) |
| Ddd | Displays the day as an abbreviation (Sun - Sat). |
| dddd | Displays the day as a full name (Sunday - Saturday) |
| dddddd | Displays the date as a complete date (including day, month, and year), formatted according to your system's short date format setting. The default short date format is m/d/yy. This option should NOT be used in file names. |
| dddddd | Displays a date serial number as a complete date (including day, month, and year) formatted according to the long date setting recognized by your system. The default long date format is mmmm dd, yyyy. |
| w | Displays the day of the week as a number (1 for Sunday through 7 for Saturday). |
| ww | Displays the week of the year as a number (1 - 53). |
| m | Displays the month as a number without a leading zero (1 - 12). If m immediately follows h or hh, the minute rather than the month is displayed. |
| mm | Displays the month as a number with a leading zero (01 - 12). If m immediately follows h or hh, the minute rather than the month is displayed |
| mmm | Displays the month as an abbreviation (Jan - Dec). |
| mmmm | Displays the month as a full month name (January - December). |
| q | Displays the quarter of the year as a number (1 - 4). |
| y | Displays the day of the year as a number (1 - 366). |
| yy | Displays the year as a 2-digit number (00 - 99). |
| yyyy | Displays the year as a 4-digit number (100 - 9666). |
| h | Displays the hour as a number without leading zeros (0 - 23). |

Value

The current date and time.

Examples

All following examples assume that it was the 25th May 1980, 04:35:00 in the morning when the command was executed:

```
!NOW:ddmmyy_hhmmss
```

results in 25051980_043500

```
!NOW:mmmm<SP>dd"th"<SP>yyyy
```

results in May 25th 1980

See Also

9.22 !POINTER

Disables or enables the pointer (blue frame) that marks the current position during a macro replay. For example, the pointer can be switched off for unattended operations where nobody watches the macro. Also, disabling the pointer can be useful to avoid losing the focus from iMacros screen. This can be necessary if you want to manually continue typing in fields without taking your hands off the keyboard while the macro is playing (e.g. during a [WAIT](#)¹¹⁴ statement).

Value

YES|NO

Examples

See Also

9.23 !REPLAYSPEED

Sets the replay speed to fast, medium or slow. Fast replay speed means there is no delay between each step, medium inserts a 1 second and slow a 2 second delay between each command. This command overrides the global replay speed setting in the options dialog.

Value

FAST|MEDIUM|SLOW

Examples

If you use iMacros for web site response measurements, remember to set the replay speed to `FAST` so that no artificial delays are added:

```
SET !REPLAYSPEED FAST
STOPWATCH ID=home
URL GOTO=http://www.iopus.com
STOPWATCH ID=home
```

See Also

9.24 !STOPWATCHTIME

Contains the last measured [response time](#)^[45] value.

Value

The last measured response time in seconds.

Examples

See Also

9.25 !TIMEOUT

Sets the timeout value in seconds. If a website, which is requested by clicking a link or using the [URL](#)^[113] command, does not load within the amount of seconds given by `!TIMEOUT`, an error is generated. Overrides the Internet macro default value. The default timeout is 60s. The [!LOADCHECK](#)^[125] value is automatically 1/10th of the timeout value.

Value

Any integer.

Examples

See Also

9.26 !URLCURRENT

Contains the current URL. This is the URL visible in the browser address bar at the time the variable is used.

Value

Any valid URL.

Examples

See Also

9.27 !URLSTART

The URL that was active in the browser when the macro started. Typically used to go back to the start page inside the macro.

Value

Any valid URL.

Examples

See Also

9.28 !VAR1

One of the three standard built-in variables for arbitrary use.

Value

Any number, character or series of characters.

Examples

```
SET !VAR1 myDearFriend
```

See Also

9.29 !VAR2

One of the three standard built-in variables for arbitrary use.

Value

Any number, character or series of characters.

Examples

```
SET !VAR2 oh<SP>I<SP>fell<SP>like<SP>setting<SP>this<SP>to<SP>:<SP>42
```

See Also

9.30 !VAR3

One of the three standard built-in variables for arbitrary use.

Value

Any number, character or series of characters.

Examples

```
SET !VAR3 23
```

See Also

9.31 !VARDEFAULT

Holds a value which is assign automatically to all undefined variables. This can be helpful for debugging a macro that it is normally used via the command line or Scripting Interface and contains many variables. With !VARDEFAULT active, you can avoid the "variable not defined" errors when running the macro manually for test purposes. If the value of !VARDEFAULT is empty, no default value is assigned.

Value

Any character or series of characters.

Examples

See Also

10 Command Line Switches

The behaviour of the iMacros Browser (included in PRO and Scripting Edition) can be controlled by so-called command line switches. These are key/value pairs added after the call to the iMacros main executable. These pairs start with a dash (-) and are separated by a whitespace.

If values include whitespaces, please use the [SP] command to escape them. Linebreaks in command line options are escaped by the [BR] command.

[datasource](#)^[134] file_name

Specifies the location and name of the datasource file.

[loop](#)^[134] max_loop

Specifies the number of times the macro will be looped.

[macro](#)^[135] macro_name

Specifies the name of the macro to replay.

[noexit](#)^[135]

Determines whether iMacro Browser is closed after replay.

[silent](#)^[135]

Specifies if the iMacros Browser is run in silent mode.

[timeout](#)^[136] seconds

Specifies the overall timeout value.

[tray](#)^[136]

Specifies if the iMacros Browser is run in tray mode.

[useragent](#)^[137] some_user_agent

Specifies the user agent.

[var varname](#)^[137] value

Assigns a value to the variable varname.

[var1](#)^[137] value

Assigns a value to the built-in variable !VAR1.

[var2](#)^[138] value

Assigns a value to the built-in variable !VAR2.

var3^[138] value

Assigns a value to the built-in variable !VAR3.

10.1 datasource

Specifies the name and location of an input file for merging data with macro. If no folder is supplied, the file is assumed to lie in the standard data input directory of your iMacros installation, e.g.

C:\Program Files\iMacros\datasources\.

If values include whitespaces, please use the [SP] command to escape them. Linebreaks in command line options are escaped by the [BR] command.

Value

Valid name and location of the input file.

Examples

```
imacros.exe -macro test1 -datasource c:\test1.csv
```

See Also

!DATASOURCE^[119], iimSet^[144]

10.2 loop

Specifies the number of times the macro will be looped. After each execution, the built-in variable !LOOP^[126] is increased by one.

If values include whitespaces, please use the [SP] command to escape them. Linebreaks in command line options are escaped by the [BR] command.

Value

Positive integer.

Examples

```
imacros.exe -macro test1 -loop 10
```

See Also

10.3 macro

Specifies the name of the macro to replay. The name must not include the file extension `.iim`. The macro has to lie in the macros folder of your iMacros installation, e.g. `C:\Program Files\iMacros\Macros\`.

If values include whitespaces, please use the `[SP]` command to escape them. Linebreaks in command line options are escaped by the `[BR]` command.

Value

Valid macro name.

Examples

```
imacros.exe -macro test1
```

will replay the `test1.iim` file in the macros folder.

See Also

10.4 noexit

If set, the iMacros Browser will not close after replaying the macro. The browser will stay open and display the last access web site.

Value

No value required.

Examples

```
imacros.exe -macro test1 -noexit
```

See Also

10.5 silent

If set, the iMacros Browser will replay the macro in silent mode. The browser window will not be seen and no icon in the Windows System Tray will indicate that iMacros is running. The process can only be seen in the Windows Task Manager.

Value

No value required.

Examples

```
imacros.exe -macro test1 -silent
```

See Also

10.6 timeout

Specifies the overall timeout value. If you start the iMacros Browser via the command line This switch is intended to watch the overall iMacros runtime and avoid any "hung" instances of iMacros. If this timeout is reached, the iMacros Browser is terminated and -320 is returned.

The default is no timeout.

Value

Any positive integer.

Examples

```
imacros.exe -macro test1 -timeout 100
```

See Also

[!TIMEOUT](#)^[130], [!MACROTIMEOUT](#)^[126]

10.7 tray

If set, the iMacros Browser will replay the macro in tray mode. The browser window will not be seen, only an icon in the Windows System Tray will indicate that iMacros is running.

Value

No value required.

Examples

```
imacros.exe -macro test1 -tray
```

See Also

10.8 useragent

Specifies the user agent. On each request to a web server, the browser sends a user agent field which identifies the browser. With this switch you can set it to anything you like, e.g. to let the web server believe that your iMacro Browser is a WAP enable mobile phone.

If the user agent string contains whitespaces, please use double quotes around it.

Value

Any character or series of characters.

Examples

```
imacros.exe -macro test1 -useragent "Nokia6230/2.0+(04.43)"
```

See Also

10.9 var_varname

Assigns a value to the variable `varname` during replay of a macro.

If values include whitespaces, please use the `[SP]` command to escape them. Linebreaks in command line options are escaped by the `[BR]` command.

Value

Any character or series of characters.

Examples

```
imacros.exe -macro test1 -var_LASTNAME Schaefer
```

will assign the value `Schaefer` to the variable `LASTNAME` in macro `test1`.

See Also

`iimSet`

10.10 var1

Assigns a value to the built-in variable `!VAR1`.

If values include whitespaces, please use the `[SP]` command to escape them. Linebreaks in command line options are escaped by the `[BR]` command.

Value

Any character or series of characters.

Examples

```
imacros.exe -macro test1 -var1 something[BR]or[SP]another
```

See Also

[!VAR1](#) 

10.11 var2

Assigns a value to the built-in variable !VAR2.

If values include whitespaces, please use the [SP] command to escape them. Linebreaks in command line options are escaped by the [BR] command.

Value

Any character or series of characters.

Examples

```
imacros.exe -macro test1 -var2 something[SP]else
```

See Also

[!VAR2](#) 

10.12 var3

Assigns a value to the built-in variable !VAR3.

If values include whitespaces, please use the [SP] command to escape them. Linebreaks in command line options are escaped by the [BR] command.

Value

Any character or series of characters.

Examples

```
imacros.exe -macro test1 -var3 Hello[BR]My[SP]name[SP]is[SP]Kurt.
```

See Also

[!VAR3](#)^[132]

11 Scripting Interface Command Overview

Available scripting commands:

All functions return an error code < 0 in case of problems. You can use [iimGetLastError](#)^[141] to retrieve the text associated with the last error. A list of return codes is available [here](#)^[145].

```
int ret_code = iimDisplay[139] ( String message [, int timeout] )
```

Displays a short message in the iMacros browser.

```
int ret_code = iimExit[140] ( [int timeout] )
```

Closes the iMacros browser.

```
String err_message = iimGetLastError[141] ( )
```

Returns the text associated with the last error.

```
String extract = iimGetLastExtract[141] ( )
```

Returns the contents of the [!EXTRACT](#)^[122] variable.

```
int ret_code = iimInit[142] ( String command_line [, boolean start_browser] [, String run_as_user, String run_as_password, String run_as_domain] )
```

Initializes the Scripting Interface.

```
int ret_code = iimPlay[143] ( String macro [, int timeout] )
```

Plays a macro.

```
int ret_code = iimSet[144] ( String var_string, String var_value )
```

Assigns values to variables.

11.1 iimDisplay

Displays a short message in the iMacros browser. A typical usage would be to distinguish several running iMacros Browsers or display information on the current position within the script. If you want to hide the display so that you can manually trigger macros with the Play button please use `iimDisplay ("#HIDEDISPLAY")`.

A list of return codes is available [here](#)^[145].

Syntax

```
int ret_code = iimDisplay ( String message [, int timeout] )
```

Parameters

String message

The message that is to be displayed in the iMacros Browser

int timeout

The optional timeout value determines when the Scripting Interface returns a timeout error if the command is not completed in time. The default value is 3 seconds.

Examples

Visual Basic Script example:

```
Dim imacros1, imacros2, iret

Set imacros1 = CreateObject("InternetMacros.iim")
iret = imacros1.iimInit()
iret = imacros1.iimDisplay("This is the 1st iMacros Browser")

Set imacros2 = CreateObject("InternetMacros.iim")
iret = imacros2.iimInit()
iret = imacros2.iimDisplay("This is the 2nd iMacros Browser")
```

See Also

11.2 iimExit

Closes the iMacros browser. It first attempts to close the browser by sending it a direct message. If the browser does not react (e.g. because the website made the browser freeze), it terminates the browser. Thus the Scripting Engine will keep control even if the browser itself is "frozen".

A list of return codes is available [here](#)^[145].

Syntax

```
int ret_code = iimExit ( [int timeout] )
```

Parameters

int timeout

The optional timeout value determines when the Scripting Interface returns a timeout error if the command is not completed in time. The default value is 3 seconds.

Examples

Visual Basic Script example:

```
Dim imacros, iret
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros.iimInit()
iret = imacros.iimPlay("mymacro")
iret = imacros.iimExit()
```

See Also

11.3 iimGetLastError

Returns the text associated with the last error.

Syntax

```
String err_message = iimGetLastError ( )
```

Parameters

None

Examples

Display a dialog if iMacros cannot be initialized (Visual Basic Script example):

```
Dim imacros, iret
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros.iimInit()
If iret < 0 Then
    MsgBox "An error occurred: " + vbNewline + _
        imacros.iimGetLastError()
End If
```

See Also

11.4 iimGetLastExtract

Returns the contents of the [!EXTRACT](#)^[122] variable. If the last command was [iimPlay](#)^[143] and if [EXTRACT](#)^[91] is used inside a macro `iimGetLastExtract` returns the extracted text. If the [EXTRACT](#)^[91] command could not find the extraction anchor, an #EANF# (Extraction Anchor Not Found) message is returned. If there is no [EXTRACT](#)^[122] command in the macro which was just played, `iimGetLastExtract` returns an empty string ("").

If in one macro several [EXTRACT](#)^[91] commands appear, the results are separated by the string

[EXTRACT]. If [complete tables](#)^[37] where extracted, adjacent table elements are separated by the string #NEXT# and ends of table rows are delimited by the string #NEWLINE#.

Syntax

```
String extract = iimGetLastExtract ( )
```

Parameters

None

Examples

Display the extracted results from a macro (Visual Basic Script example):

```
Dim imacros, iret
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros.iimInit()
iret = imacros.iimPlay("myextractmacro")
MsgBox "The extract was: " + vbNewline + _
    imacros.iimGetLastExtract()
iret = imacros.iimExit()
```

See Also

11.5 iimInit

Initializes the Scripting Interface and opens a new instance of the iMacros Browser unless otherwise stated by the `command_line` parameter. If you use the Run As feature of the `iimInit` command, you must use the `SetRunAs.exe` utility to store the user name and password of the Run As user in the registry. The tool can be found in the iMacros Program directory (typically `C:\Program Files\iMacros\`). Experts can also change the settings manually using the `dcomcnfg` utility from Microsoft, but the use of the provided tool is recommended. If you do not make this required setting, `iimInit` will notify you by returning a -5 error code.

A list of return codes is available [here](#)^[145].

Syntax

```
int ret_code = iimInit( String command_line [, boolean start_browser] [,
String run_as_user, String run_as_password, String run_as_domain] )
```

Parameters

String command_line

Specifies command line switches. All switches start with a dash (-). Possible switches are:

- ie Starts the Internet Explorer instead of the iMacros Browser.
- tray Starts the iMacros Browser in tray mode.
- silent Starts the iMacros Browser in silent mode, i.e. tray mode with a tray icon.

boolean start_browser

Specifies whether to connect to an open instance of the iMacros Browser or Internet Explorer. If set to false, iMacros tries to connect to an existing instance. If none is found, a new instance is created. Default is true.

String run_as_user

Username under whose account the iMacros Broser is started.

String run_as_password

Password needed for the account specified in run_as_user.

String run_as_domain

Domain in which the username given in run_as_user is valid.

Examples

Initialize the Scripting Interface in silent mode (iMacros not visible in taskbar or tray - Visual Basic Script example):

```
Dim imacros, iret
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros1.iimInit("-silent")
```

Initialize the Scripting Interface connection to an existing Internet Explorer running under a certain user(Visual Basic Script example):

```
Dim imacros, iret
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros1.iimInit("-ie", FALSE, "username", "password", "domain")
```

See Also

11.6 iimPlay

Plays a macro. After the macro has played all options that have been set with the iimSet command are reset. Use [iimGetLastExtract](#)^[141] to get [extracted text](#)^[30].

There are two fundamentally different ways of playing a macro using the iimPlay command. The first is to specify the filename (without the extension) of the macro in the String macro parameter. The other is to generate macro code on-the-fly in your program and passing it, preceded by CODE: directly to iimPlay using the String macro parameter. Several commands in a macro generated on-the-fly must be separated by the CR symbol. These are vbNewLine or vbCrLf in Visual Basic or \r\n in C, C++ or C#.

If you start a macro via iimPlay which contains a TAG TYPE=SELECT... statement and the specified value is not in the drop down list, the iimPlay command returns an -251 error code. In the corresponding error message (see [iimGetLastError](#)^[141]) the maximum index is given. You can use this value, for example, to always select the last entry of a changing drop down list.


A list of return codes is available [here](#)^[145].

Syntax

```
int ret_code = iimPlay ( String macro [, int timeout] )
```

Parameters

String macro

Either the macro's filename without the extension or a string holding [macro commands](#) 

int timeout

The optional timeout value. If the command is not completed during this time span, the Scripting Interface returns a timeout error -2. No extraction data is returned in this case. The default value is 600 seconds.

Examples

Play a macro located in the Macros\ directory of your iMacros installation (Visual Basic Script example):

```
Dim imacros, iret
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros.iimInit()
iret = imacros.iimPlay("mymacro")
```

Play some on the fly generated code (Visual Basic Script example):

```
Dim imacros, iret, mycode, myURL

myURL = "http://www.iopus.com"

mycode = "URL GOTO=" + myURL + vbNewLine
mycode = mycode + "TAG POS=1 TYPE=FONT ATTR=TXT:<SP><SP>Online<SP>Store"

Set imacros = CreateObject("InternetMacros.iim")
iret = imacros.iimInit()
iret = imacros.iimPlay("CODE:" + mycode)
```

See Also

11.7 iimSet

Assigns values to variables. There are limitations as to what variables you can set using this command. You can set all built-in variables *which you also can set via the command line*. Additionally, you can set all user defined variables. After `iimPlay` all variables are erased. The return code is always 0.

Syntax

```
int ret_code = iimSet ( String var_string, String var_value )
```

Parameters

String var_string

A string defining which variable is to be set. Use `-var_varName` for any user defined variable named `varName`. Use the following `var_string` for the built-in variables:

| | |
|--------------------------|---|
| <code>-timeout</code> | <code>!TIMEOUT</code> ^[130] |
| <code>-loop</code> | <code>!LOOP</code> ^[126] |
| <code>-datasource</code> | <code>!DATASOURCE</code> ^[119] |
| <code>-var1</code> | <code>!VAR1</code> ^[131] |
| <code>-var2</code> | <code>!VAR2</code> ^[131] |
| <code>-var3</code> | <code>!VAR3</code> ^[132] |

String var_value

The value which is to be assigned to the variable.

Examples

Loop over a number, for example to extract one table element after the other

```
Dim imacros, iret, i
Set imacros = CreateObject("InternetMacros.iim")
iret = imacros.iimInit()
For i=0 To 4
    ' You have to convert the value into a string!
    iret = imacro.iimSet("-var_loop", CStr(i))
    iret = imacros.iimPlay("mymacro")
Next
```

See Also

11.8 Scripting Interface Return Codes

After each command the Scripting Interface returns a code. You can use this return code to find out whether a command succeeded or not. The general rule is:

| | |
|---------|-----------------------------|
| Success | return code greater equal 0 |
| Failure | return code less than 0 |

You can use `iimGetLastError`^[141] to retrieve the text associated with the last error.

1 (sOk)

Macro completed Ok

-1 (sFail)

Scripting Interface can not complete this command. Typically this error can occur if the software is not installed correctly

-2 (sNotCompleted)

A command was started but did not complete. Typically this happens if the user manually closes the browser while a macro is running.

-3 (sTimeout)

The iMacros browser did not respond in a certain time. The default timeout is 600s. You can change this value individually [for some commands](#)^[139].

-4 (sNotStarted)

Could not start the iMacros browser.

-5 (sCheckRunAs)

Could not reach the iMacros browser. Windows security requires that if you use the "Run As" feature of the [iimInit](#)^[142] command, you must use the "SetRunAs" utility to store the user name and password of the "Run As user" in the registry.

-100...-999 (sMacroError)

sMacroError can have a value of -100 and lower (-101,-102,...,-999). These are NOT Scripting Interface error codes, but errors created by the iMacros browser and the macro itself. The Scripting Interface passes them from the browser to your script or application.

-101

User pressed Stop button in the iMacros Browser. Typically, you can check on this value to see if the user wants to exit the application.

-102

User pressed Exit button in the iMacros Browser, i.e. the iMacros Browser was closed by a user. Note that this is **not** the same as sNotComplete. The code "sNotComplete"(-2) indicates that the browser was closed e.g. by the task scheduler but not by a regular user exit or stop.

Error codes **below -200** are error codes generated during the macro replay. These are the same error codes that you get during a manual macro replay. For more information please see the [List of IIM Browser Error Codes](#)^[48].

12 How to buy iMacros

Direct order link

<http://www.iOpus.com/store>

Available payment methods: Credit card, check, money order, purchase order (PO), bank/wire transfer, Switch/Solo, Cash

Product Home page

<http://www.iOpus.com/imacros>

Email Sales: <http://www.iopus.com/service/contact>

Email Support: <http://www.iopus.com/service/support>

User forum: <http://forum.iopus.com>

13 Feature comparison

Compare the features of the different iOpus iMacros Editions.

A detailed feature comparison list can be found online at:

<http://www.iopus.com/iim/compare>

If you are unsure which version is right for you, please ask us at support2@iopus.com .

Index

- ! -

!COL 119
 !DATASOURCE 119
 !DATASOURCE_COLUMNS 120
 !DATASOURCE_LINE 120
 !DIALOGMANAGER 120
 !ENCRYPTION 121
 !ERRORIGNORE 121
 !ERRORMACRO 122
 !EXTRACT 122
 !EXTRACT_TEST_POPUP 123
 !EXTRACTADD 123
 !EXTRACTDIALOG 123
 !FILELOG 124
 !FILESTOPWATCH 124
 !FOLDERIMACROS 124
 !IMAGEX 125
 !IMAGEY 125
 !LOADCHECK 125
 !LOOP 126
 !MACROTIMEOUT 126
 !NOW 127
 !POINTER 129
 !REPLAYSPEED 129
 !STOPWATCHTIME 130
 !TIMEOUT 130
 !URLCURRENT 130
 !URLSTART 131
 !VAR1 131
 !VAR2 131
 !VAR3 132
 !VARDEFAULT 132

- # -

#NEWLINE# 38
 #NEXT# 38

- % -

%ERRORLEVEL% 50

- . -

.iim 12

- / -

/DIR 66
 /NOCANCEL 66
 /SILENT 66
 /VERYSILENT 66

- [-

[BR] 50
 [EXTRACT] 38
 [SP] 50

- { -

{{Variable}} 25

- < -

 25, 79
 <SP> 25

- 7 -

7 x 24 operation 82

- A -

Access 71
 Active Directory 67
 ActiveX Component 65
 Ad Blocking 19
 Add 71, 87
 AES 13
 Alert Dialog 19
 Asian Characters Extraction 41
 Assert Keyword 79
 Assert Keyword With EXTRACT 80
 Assert Keyword With IMAGESEARCH 81
 Assert Keyword With TAG 79

Assign Variables 25

Automation 49

- B -

BACK 88

BAT 50

Batch Files 50

Blank Tab 22

BMP 61

Built-in Variables 25, 116

Buy Internet Macros 146

Bypass 26

- C -

C++ 71

Calculations 71

CAPTCHA 71

Certificate Dialog 20

Change Editor 12

Change User Agent 46

Changing URL 70

Chinese Characters Extraction 41

CLEAR 88

CLICK 89

Click (Mouse) 42

Click Images (IRP) 60

Click Mode 11

Client Authentication 20

CMDLINE 90

Comma Separated Values 27

Command Line Switches 132

Comment 91

Conditional Statements 68

Confirmation Dialog 19

CPU Usage 82

Create Shortcuts 49

Credit card 146

- D -

Data Merging 27

Database 71

■ ■ ■

-datasource 134

DEBUG 102

Deployment 65, 66, 69

Dialog Manager 18

Dialog Not Managed 75

Disable images 47

DISCONNECT 91

Display Variable Content 73

Double-Byte Support 41

Download Files 16, 17

Download Gallery 16, 17

Download Image 16, 17

Download Picture 16, 17

Download Target 17

Download Thumbnails 16, 17

- E -

Early Binding 55

Edit 12

Editions 147

Editor 12

Email Notifications 47

Encryption 13, 121

Error Codes 48, 145

Error handling 48

ERRORLEVEL 50

EVENT:MOUSEOVER 43

Excel 71

EXTRACT 91

Extract Complete Tables 37

Extract Complete Websites 37

Extract Current URL 79

Extract Data 30

Extract Data Using Scripting Interface 38

Extract Dialog Text 78

EXTRACT Error 23

Extract Hidden Input Fields 78

Extract Results 76

Extract Separate Lines 79

Extraction Anchor 35, 40

Extraction Anchor Not Found Message 75

Extraction of <PRE> 30

Extraction Tips 40

Extraction Wizard 30

- F -

FAQ 68

Features 147

FILEDELETE 93

Fill Extracted Data 76

Fill Forms 27

Fill Hidden Input Fields 78

FILTER 94

Filter Images 47

FireFox 46

Flash Applications Response Time 60

Flash Chat Applet (IRP) 64

Foxpro 71

FRAME 94

Frames 23

- I -

Icon 69

If / Then 68

iim.ini 67

iimDisplay 139

iimExit 140

iimGetLastError 141

iimGetLastExtract 141

iimInit 142

iimPlay 143

iimRunner 57, 59

iimSet 144

iMacros Browser 9

imacros.dll 65

imacros.exe 65

Image Creation for IRP 61

Image Recognition 60

Image Search 60

IMAGECLICK 95

IMAGESEARCH 96

imatl.dll 65

imgr.dll 65

imsys.dll 65

Input Box Not Recorded 74

Input Data 27

Input From Access 29

Input From CSV files 27

Input From Database 29

Input From List Of Variables 28

Input From MDB 29

Input From MS SQL 29

Input From MySQL 29

Input From Oracle 29

Input From SAP 29

Input From SQL 29

INPUT TYPE=FILE 42

Install 69

Installation 66, 67

Installer 69

Instr 38

Intellisense 55

Internet Explorer 9, 69

Internet Explorer Compatibility 68

Internet Explorer Plug-In 9

Internet Monitoring 82

internetmacros.dll 65

- J -

Japanese Characters Extraction 41

Java 71

Java Chat Applet (IRP) 64

Javascript Dialogs 19

JavaScript Error 21

Javascript Menu 43

- K -

Keystrokes 42

Keyword Extraction 30

Keyword Search 79

Keyword Search With EXTRACT 35, 80

Keyword Search With IMAGESEARCH 81

Keyword Search With TAG 79

Korean Characters Extraction 41

- L -

Late Binding 55

Link Macros Together 72

List Of Variables 28

Lnk 49

Load Test 44

Local Pages 22

Login Dialog 18

- - -

-loop 134

Loop Inside Macro 73

-macro 135

- M -

Memory leak 82

Memory Usage 82

Modify User Settings Directly 67

Monitoring 82

Mouse Clicks 42

MouseMove 43

MouseOver 43

MS Paint 61

MySQL 71

- N -

NEWLINE 38

NEXT 38

- - -

-noexit 135

Nokia 46

Non-stop operation 82

Notepad 12

- O -

Offline 22

On Error Continue 71

ONCERTIFICATEDIALOG 97

ONDIALOG 97

ONDOWNLOAD 98

ONERROR 99

Online Store 146

ONLOGIN 99

ONPRINT 100

ONSECURITYDIALOG 101

ONWEBPAGEDIALOG 101

Open New Window 22

Open Tab 22

Opera 46

- P -

Page Error 21

Page Loading 74

Passwords 13

PAUSE 102

Payment options 146

Performance 44

Perl 71

Player 65

Player License 69

Plugin 9

Pocket PC 46

Popup Blocking 19

Popups 19

Position Parameter in EXTRACT 33

Power Surfer Edition 147

PRINT 103

Print Dialog 20

Print Frames 21

Print Web Page 20, 21

PRO Editon 147

PROMPT 103

Prompt Dialog 19

PROXY 104

Proxy Server 26

Purchase Internet Macros 146

Purchase order 146

- R -

Radio Button 72

Random Numbers 71

Recording 11

Recording Speed 11

REDIAL 105

Redistributing iMacros 65, 69

REFRESH 106

Relative Extraction 34

Relative POS 110

Replay 12

Replay Loops 12

Replay Speed 12

Response Time Measurement 44

Response Time Measurement Tips 45
 Response Time Measurements 82
 Response Time Measurements Example 45
 Restricted User 67
 Royalty free license 65
 Run As 52, 57
 Run at specific time 52
 RunAs 57, 59

- S -

Save Data To Access 38
 Save Data To Database 38
 Save Data To MDB 38
 Save Data To MS SQL 38
 Save Data To MySQL 38
 Save Data To Oracle 38
 Save Data To SAP 38
 Save Data To SQL 38
 Save Extracted Data 38
 Save Item 16
 Save Picture As 17
 Save Target As 17
 Save Types 14
 Save Web Page 14
 Save Web Page Elements 16
 SAVEAS 106
 Schedule Tasks 52
 Scheduling 52
 Screen Scraping 30, 37
 Screenshots 14
 Script Error 21
 Scripting Edition 147
 Scripting Interface 53
 Scripting Interface Error Codes 145
 Search Images (IRP) 60
 Security 13
 Security Dialog 20
 Send Email 47
 SET 107
 Set Focus 71
 Setup 65, 66
 Shortcuts 49
 ShowModalDialog 19
 ShowModelessDialog 19

■ ■ ■

-silent 135
 Simulate Internet Explorer 75
 SIZE 108
 Split 38
 Split Extracted Data 36
 SQL 71
 SrvAny 57
 Start 9
 Start From Web Page 58
 STOPWATCH 109
 Stress Test 44
 Submitting Data 27
 Subtract 71

- T -

TAB 110
 Tabbed Browser 22
 TAG 110
 TAG Editing 24
 TAG Error 23
 TAG Tuning 24
 Taking Screenshot 14
 Task Manager 82
 Task Scheduler 52
 Tell me about 8
 Test If Image Loaded 81

■ ■ ■

-timeout 136
 Timer 52
 Toolbar 69
 -tray 136
 Trigger Mouse Event 43

- U -

Unsupervised operation 82
 URL 113
 User Agent 46
 User Input 72
 User Interface 9
 User Settings 67

■ ■ ■

-useragent 137
User-defined Variables 25
-var_varname 137
-var1 137
-var2 138
-var3 138

- V -

Variables 25
VB.NET 55
VB6 55
VBA 71
VERSION 114
Visual Basic Example 55
Visual Basic Script Example 54

- W -

WAIT 114
Web Page Dialog 19
Web Scraping 30, 37
Website Compatibility 68
Welcome 7
Wildcards In EXTRACT 35
Wildcards In TAG 24
WINCLICK 115
Windows Mobile 46
Windows Scripting Host 53
Windows Service 57
Work Offline 22
WSH 53

- Z -

ZIP compression 14