

# HTTPFetch

version 2.3.x - revision 6

## Administrator's Guide

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express permission of Autonomy.

Windows is a trademark of Microsoft Corp.

Unix is a trademark of X/Open Ltd.

Copyright © 2002 Autonomy

HTTPFetch is a trademark of Autonomy

# Table of Contents

<b>1. Introduction.....</b>	<b>1</b>
System architecture .....	2
<b>2. Installing HTTPFetch.....</b>	<b>3</b>
System requirements .....	3
Installing HTTPFetch under Windows NT or 2000 .....	5
Directory structure: Windows .....	8
Installing HTTPFetch under UNIX.....	10
Directory structure: UNIX.....	12
<b>3. Getting Started.....</b>	<b>13</b>
Adding a new site.....	13
Running the process .....	20
Using the Login settings for secure sites .....	21
Using NTLM Proxy to access sites that use NTLM authentication .....	22
<b>4. Configuring HTTPFetch .....</b>	<b>23</b>
Configuration file sections .....	24
[License] section .....	24
[Service] section .....	25
[Default] section.....	26
[<MySpider>] section .....	56
Typical Configurations.....	56
Example HTTPFetch configuration file .....	57
<b>5. HTTPFetch Administration Utility .....</b>	<b>59</b>
Starting the Administration Utility .....	59
Selecting an alternative instance of the HTTPFetch to configure.....	59
The Main Window .....	60
The Startpoint Page .....	64
The Limits Page.....	67
The Page Criteria Page.....	70
Import Parameter Settings.....	72
The Date Page.....	94
Site.....	96
Directory Specification.....	97
The Links Page.....	98
The Link pre/post Page .....	100

The Links start/end Page.....	101
The Wildcards Page.....	102
The Login Page.....	103
The Advanced Page.....	105
The Site Statistics Page.....	107
The Site Map Page.....	111
<b>6. HTTPFetch Tutorial .....</b>	<b>113</b>
Spidering .....	113
Scheduling.....	117
Downloading Pages .....	118
Importing.....	125
Example Configuration .....	127
Running the Fetch.....	129
Manipulating HTML Rendered files .....	134
<b>7. HTTPFetch Troubleshooting.....</b>	<b>135</b>
Why is the HTTPFetch not fetching any data? .....	135
Why has the HTTPFetch not managed to index the data?.....	135
Why does the HTTPFetch get so many uninteresting pages? .....	136
Why does the HTTPFetch stop spidering half way through?.....	136
Why are documents not being imported/indexed?.....	136
Why is my spider downloading so many pages each time it runs?.....	137
Why does the HTTPFetch fail to access certain web sites?.....	137
<b>8. HTTPFetch FAQ.....</b>	<b>139</b>
<b>Appendix A: Service settings for Autonomy DiSH.....</b>	<b>147</b>
<b>Appendix B: The NTLM Proxy module .....</b>	<b>149</b>
Installing the NTLM Proxy module.....	149
Finding out whether a site uses NTLM authentication .....	150
Pointing the HTTPFetch to the NTLM Proxy module.....	151
Configuring the NTLM Proxy module.....	152
<b>Appendix C: Supported date formats for spidering .....</b>	<b>153</b>
<b>Appendix D: Error messages.....</b>	<b>155</b>
<b>Glossary.....</b>	<b>161</b>
<b>Index.....</b>	<b>163</b>



## Autonomy

---

Autonomy employs a fundamentally different and unique combination of technologies to enable computers to form an understanding of a page of text, web pages, e-mails, voice, documents and people.

Autonomy's solution is therefore able to power any application dependent upon unstructured information within every market sector, including: e-commerce, customer relationship management, knowledge management, enterprise information portals and online publishing applications.

This is evidenced by the significant penetration of the technology in a diversity of vertical markets and has been achieved principally because every market sector needs to manage and leverage the benefits of unstructured information.

Autonomy was founded in 1996 and has offices in Boston, Chicago, Dallas, San Francisco, New York, and Washington, D.C. in the United States, as well as offices throughout EMEA, including Amsterdam, Brussels, Cambridge, Frankfurt, Milan, Paris, Oslo, and Sydney. In July 1998, the company went public on the EASDAQ exchange (EASDAQ:AUTN). Autonomy floated on The NASDAQ National Market (NASDAQ: AUTN) in May 2000, and on the London Stock Exchange (LSE: AU.) in November 2000.

To contact Autonomy, please get in touch with your nearest location listed below.

### **Europe and South Pacific**

Autonomy Systems Ltd.  
Cambridge Business Park  
Cowley Road  
Cambridge  
CB4 0WZ

Help Desk: +44 (0) 800 0 282 858

Switchboard: +44 (0) 1223 448 000

Fax: +44 (0) 1223 448 001

E-mail for information: [autonomy@autonomy.com](mailto:autonomy@autonomy.com)

for support: [uksupport@autonomy.com](mailto:uksupport@autonomy.com)

The Help Desk operates from 9.30 am to 6.00 pm (GMT) Monday to Friday.

Website: [www.autonomy.com](http://www.autonomy.com)

### **USA**

Autonomy Inc.  
301 Howard Street  
22<sup>nd</sup> Floor  
San Francisco  
CA 94105

Help Desk: +1 877 333 7744

Switchboard: +1 415 243 9955

Fax: +1 415 243 9984

E-mail for information: [info@us.autonomy.com](mailto:info@us.autonomy.com)

for support: [support@us.autonomy.com](mailto:support@us.autonomy.com)

The Help Desk operates from 9.30 am to 6.00 pm (CST) Monday to Friday, toll-free.

Website: [www.autonomy.com](http://www.autonomy.com)

# Welcome

---

Thank you for choosing Autonomy and welcome to your HTTPFetch™ version 2.3 Administrator's Guide.

## Autonomy Solutions

Autonomy solutions provide the software infrastructure that automates operations on unstructured information. This software infrastructure is based on IDOL server, the Intelligent Data Operating Layer. IDOL makes it possible for organizations to process digital content automatically and to enable applications to operate with each other. It consists of data operations that integrate information by understanding any type of content, and is therefore data agnostic. The IDOL server software infrastructure is fully scalable and customizable according to customers' present and future needs.

Autonomy solutions include:

- **Autonomy Connectors™** enable automatic content aggregation from any type of local or remote repository (for example, a database, a web site, a real-time telephone conversation etc.), facilitating a unified solution across all information assets within the organization.
- **ACI Servers™** automatically perform a variety of operations on structured, unstructured and semi-structured information (documents, audio, video etc.). These operations include automatic hyperlinking, tag reconciliation, XML tagging, profiling, alerting, categorization, cluster mapping, real-time transcription, targeting etc.
- **Autonomy Application Builder™** is a toolkit that enables companies and partners to customize Autonomy's products according to their individual requirements. It facilitates easy communication between custom-built applications that retrieve data using HTTP commands and the Autonomy ACI servers, as well as simple manipulation of the returned result sets. Communication with the servers is implemented over HTTP using XML and can adhere to SOAP. The API is distributed with a set of sample code.
- **Portal-in-a-Box™**, our comprehensive and fully automated Information Portal for content-rich Internet and Intranet sites.
- **Portlets** are windows that can be set up in Autonomy's Portal-in-a-Box or third party Portals. Each portlet contains an application that allows the Portals' end users to benefit from a variety of IDOL server functionality.
- **Autonomy Desktop Suite™** brings the power of Autonomy to every desktop. Conducting a real-time analysis of the ideas involved in the content of any opened desktop application, Desktop Suite's ActiveKnowledge or Active Windows Extensions module provides real-time links to relevant internal and external information without the user being needlessly diverted from their work in progress to perform an exasperating search or retrieval operation.
- **Autonomy Product Orientated Drop-in Solutions™** allow Autonomy solutions to be easily integrated with third party applications and solution providers. PODS enable organizations to make their existing applications compatible with IDOL with minimal configuration and administration requirements. Making IDOL server a part of any solution delivers the direct benefits of content automation and the ability to perform a vast range of IDOL server operations, irrelevant of file format or location.



# 1. Introduction

---

HTTPFetch is a powerful “spidering” solution allowing documents from Internet or Intranet sites to be gathered from remote WWW site servers and indexed into an Autonomy DRE (Dynamic Reasoning Engine).

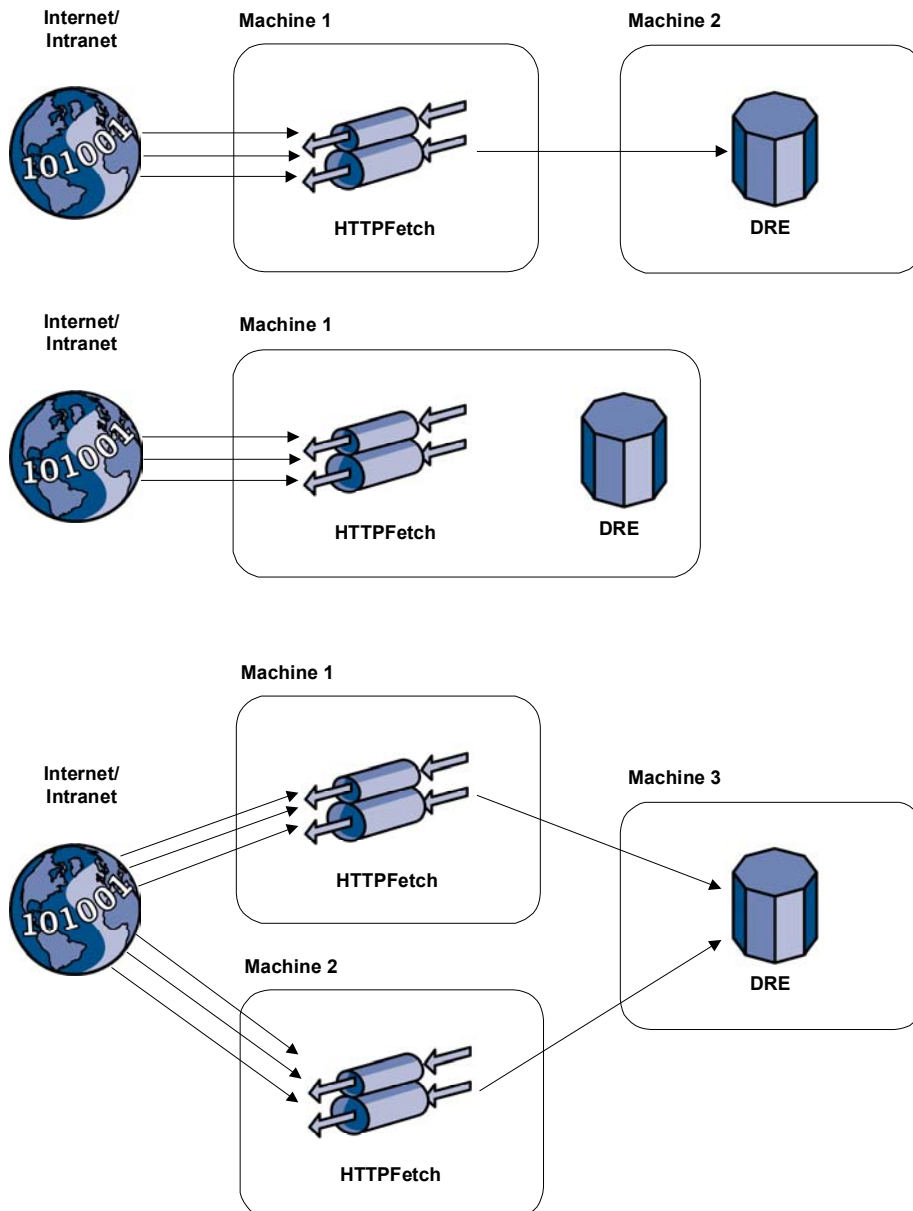
HTTPFetch uses HTTP requests to gather documents from remote WWW sites. Once it has obtained a document it automatically analyses the document for links to other documents. These links are followed conditionally based on the settings specified in the configuration and further documents retrieved. The documents retrieved are then automatically imported into Autonomy's indexing file format and indexed. The process of gathering information from a given site is referred to as **spidering** and the entire process:- spidering, importing and indexing, is referred to as a **fetch**.

Each instance of HTTPFetch can “spider” multiple websites simultaneously, making efficient use of processor and bandwidth resources.

The product has been developed to be highly configurable and to allow many different possible architectures. This ensures that HTTPFetch can be tuned and scaled to the application's exact requirements.

## System architecture

Illustrated below are three architectures in which the HTTPFetch could be set up.



## 2. Installing HTTPFetch

---

HTTPFetch can be installed in a number of ways. The simplest is to install a single HTTPFetch on a single machine, and configure it to gather documents from a number of WWW or Intranet sites. In this case the HTTPFetch application will read its configuration from a single file, present in the same directory as the application. The installer installs an example configuration file. If you wish to add sites to the configuration file you will have to do this once the installation is completed.

### Installing HTTPFetch with Portal-in-a-Box

If you are installing Portal-in-a-Box you do not need to install HTTPFetch individually as it is automatically installed with Portal-in-a-Box.

### System requirements

Minimum suggested server requirements

Windows NT and 2000 (I tel)	Solaris 2.5 (SUN SPARC)
200 MHz Pentium processor	128 MB of RAM
64 MB RAM	2 GB hard disk recommended
2 GB hard disk recommended	

**Note:** this specification is dependent on the amount of data to be fetched. Due to substantially different disk usage patterns it is beneficial to run fetch and DRE processes on separate drives or partitions.

**Before installing HTTPFetch you should have the following information ready:**

<b>Installation Name</b>	If you are licensed to have multiple installations, then each installation requires a distinct name to differentiate it from other installations.
<b>License Holder Name</b>	The name of the person or company to which the product is licensed. (For evaluation version only). If you obtain a full license key later, you simply have to edit the configuration file and change the HOLDER=, KEY= and FULL= keys.
<b>Firewall/Proxy server settings</b>	If a firewall or proxy server is between the machine you wish to run HTTPFetch on and the Internet, then you will require these settings in order to gather documents from remote WWW sites. These settings include the <b>IP Address</b> , <b>port</b> of the firewall/proxy, the <b>username</b> and <b>password</b> you use to gain access.
<b>Autonomy Query Engine settings</b>	HTTPFetch will automatically index documents into your Autonomy Server or Update. You require the settings for the DRE you wish to index the data into, these include: the <b>IP Address</b> of the DRE machine, the <b>Query</b> and <b>Index Port</b> (this can be found listed in the DRE.INI file in the main directory of your DRE), the <b>Database</b> you wish to index the documents into (again, the possible databases are listed in your DRE.INI file)
<b>Indexing Details</b>	HTTPFetch uses a file to store indexing information, the DRE is then instructed to index in this file. You need to decide where you wish these files to reside, and you will need to know the full path of the directory from the HTTPFetch and from the engine.
<b>Socket Details</b>	HTTPFetch can be set up to make use of many sockets. The maximum number of sockets you wish it to use can be setup at install time (and later changed). This should be set based on the bandwidth you have available from the machine, and the number of other socket based applications running on the machine. For example, if your main query engine (DRE) is running on the same machine it is advisable to reduce the number of sockets used by HTTPFetch, thereby allowing more responsive querying. If you only have a low-bandwidth link to the internet, it is also advisable to reduce this number. In low bandwidth situations 16 is a sensible number, if a high bandwidth is available it can be increased up to 256 or more.
<b>Sites to gather documents from</b>	It is advisable to have a list of sites you wish to gather documents from ready at install time. The installer will allow you to configure these sites (NT only) and will write the configuration file for you. If you do not have this information or wish to add or change sites at a later point then you will need to edit the configuration file.

## Installing HTTPFetch under Windows NT or 2000

To install under Windows insert the HTTP Fetch CD-ROM into your CD-Rom Drive, and start the installation by double-clicking on the **HTTPfetch-2.3.x.exe** program in the root directory of the CD-ROM through Windows explorer.

Read and follow all installation instructions on the screen carefully.

1. The installation opens with the **Welcome** dialog. Read the text and click on **Next**.
2. The **Autonomy HTTP Fetch License Agreement** dialog is displayed. Read the license agreement and click on **Next** to accept it.

If you are installing the HTTP Fetch 30 day evaluation version, the **Autonomy HTTP Fetch Limited Evaluation License** dialog is displayed. Read the license agreement and click on **Next** to accept it.

3. If you are installing the HTTP Fetch 30 day evaluation version, the **License Holder Details** dialog is displayed. Enter the license holder and click on **Next**.

**Note:** If you have purchased a fully enabled version of HTTP Fetch, check if the installation CD-ROM contains the licensekey.txt file, and copy this file into the same directory as the HTTP Fetch installer. Contact Autonomy Technical Support if you are missing the license key.

4. The **Installation Name** dialog is displayed.

Enter a unique name for the HTTP Fetch installation, and click on **Next**. Note that the unique name must not contain any spaces.

5. The **Choose Destination Location** dialog is displayed.

Select the directory in which you want to install HTTP Fetch, and click on **Next**. By default HTTP Fetch is installed on **C:\Autonomy\<InstallationName>**, but you can use the **Browse** button to navigate to another location.

6. The **Select Components** dialog is displayed.

Specify which HTTP Fetch components you want to install by checking the appropriate boxes, and click on **Next** when you are finished. The following components are available:

### **Fetch Service & example configuration file**

The main HTTP Fetch component. You must check this box for the Fetch to be able to operate.

### **Fetch Administration Utility**

A utility for administering and configuring HTTP Fetch.

### **Fetch Administration Online Help**

Online help information for the Administration Utility.

You can view how much disk space the components require and how much disk space you have left at the bottom of the dialog.

7. The **DiSH Settings** dialog is displayed. Enter the following details, and click on **Next**:

**DiSH Service Port**

Enter the port number that HTTP Fetch will use for DiSH communication. **Note:** This port must not be used by any other service.

**Service Control Clients**

Enter the IP address of machines that are permitted to control the HTTP Fetch service.

**Note:** If you want to permit a number of machines to control the HTTP Fetch service, you can use a wildcard in the IP address.

For example: enter **187.\*.\*** to permit any machine whose IP address begins with 187 to control the HTTP Fetch service.

**Service Status Clients**

Enter the IP address of machines that are permitted to access the HTTP Fetch status but are not permitted to control it.

**Note:** If you want to permit a number of machines to access the HTTP Fetch status, you can use a wildcard in the IP address.

For example: Enter **187.\*.\*** to permit any machine whose IP address begins with 187 to access the HTTP Fetch status.

8. The **Proxy Settings** dialog is displayed.

If you use a firewall or proxy server to access the Internet, enter the following details for the proxy server, and click on **Next**:

**Proxy Server**

Enter the IP address (or name) of the machine on which the proxy server is located.

**Proxy Port**

Enter the port used by the proxy server.

**Proxy Username**

Enter your username for the proxy server.

**Proxy Password**

Enter your password for the proxy server.

9. The **DRE Settings** dialog is displayed:

Enter the following details for the DRE into which you want to index the documents that HTTP Fetch retrieves:

**DRE Hostname**

Enter the IP address (or name) of the machine on which the DRE is running.

**Query Port**

Enter the port number that is used to send queries to the DRE.

**Index Port**

Enter the port number that is used to index documents into the DRE.

10. The **User Agent Settings** dialog is displayed.

Enter some contact information, such as an e-mail address, so that the site that HTTP Fetch spiders can contact you.

11. The **Indexing Details** dialog is displayed.

Enter the following details for the indexing process, and click on **Next**:

**DRE reads index files from**

Enter the path to the folder from which the DRE reads the index files.

**Database**

Enter the name of the DRE database into which you want HTTP Fetch to index the documents that it fetches.

12. The **Start Installation** dialog is displayed.

Click on **Next** to confirm the settings that you have made and to start the installation. Alternatively, click on **Back** to return to previous dialogs and make the appropriate changes.

13. The **Installing** dialog is displayed.

The progress of the installation process is indicated. If you want to abort the installation process, click **Cancel**.

14. The **Start Menu Shortcuts** dialog is displayed.

Specify whether you want to add HTTP Fetch shortcuts to the **Start** menu and click on **Next**.

15. The **Services** dialog is displayed.

Specify whether you want the HTTP Fetch service to be started immediately after the installation is complete, and click on **Next**.

16. The **Installation Complete** dialog is displayed.

HTTP Fetch has been installed successfully. Click on **Finish** to exit the installation.

You can now edit the HTTP Fetch configuration file (which is located in the **C:\Autonomy\<InstallationName>** folder) and start the HTTP Service.

## Directory structure: Windows

Once the installation of HTTP Fetch is completed, your installation directory will contain the following files:

<b>Dat</b>	Folder that contains internal data files.
dre4.dat	Data file that contains the configuration values and keys for the DRE.
httpfetch.dat	Data file that contains the configuration values and keys for HTTP Fetch.
mailer.dat	Data file that contains the configuration values and keys for Mailer.
license.dat	Internal license data file.
service.dat	Internal service data file.
AutonomyServiceManager.cfg	Configuration file that contains the settings for Autonomy Service Manager.
AutonomyServiceManager.chm	Autonomy Service Manager online help system.
AutonomyServiceManager.exe	Autonomy Service Manager executable.
AutonomyServiceManagerInstall.log	Installation log file that lists the details of the installation process for Autonomy Service Manager.
AutonomyServiceManagerUnwise.exe	Executable to uninstall the Autonomy Service Manager from your computer.
binslave.exe	Executable that parses binary files looking for ASCII content, which can be imported into the DRE.
Various DLL files	Filters used by Omnislave to convert various document types to HTML format.
importslave.exe	Executable that generates IDX files for the DRE.
ImportWizard.exe	Plugin for the HTTP Fetch Admin application, which controls how files are imported.
Install.log	Installation log file that lists the details of the installation process.
omnislave.cfg	Omnislave configuration file that contains the Omnislave settings.
omnislave.exe	Omnislave executable that parses files not in HTML or PDF format to IDX files.



pdfslave.exe	Executable that parses PDF files to IDX files.
phraselist.dat	Data file used by Bin Slave.
phraselist_doc.dat	Data file used by Bin Slave.
phraselist_ppt.dat	Data file used by Bin Slave.
phraselist_qxd.dat	Data file used by Bin Slave.
<InstallationName>.cfg	Configuration file that contains the HTTP Fetch settings.
<InstallationName>.exe	HTTP Fetch executable.
<InstallationName>Admin.cfg	Configuration file for the HTTP Fetch Admin application.
<InstallationName>Admin.exe	Executable for the HTTP Fetch Admin application.
<InstallationName>Admin.log	Log file for the HTTP Fetch Admin application.
Various TXT files	Text conversion tables used during the importing process.
Unwise.exe	Executable to uninstall HTTP Fetch from your computer.

## Installing HTTPFetch under UNIX

To install under Unix insert the HTTP Fetch CD-ROM into your CD-Rom Drive. Read and follow all installation instructions on the screen carefully.

1. Copy the HTTP Fetch installer from the CD to your local disk.
2. Uncompress the installer using the command:

```
uncompress <Installer>.tar.Z
```

3. Un-tar the resulting file using the command:

```
tar -xvf <Installer>.tar
```

This creates a subdirectory called **httpfetch-2.3.x**, which contains the following files:

```
Setup.sh
Various TXT files
Uninstall.sh
binslave.exe
freplace.exe
httpfetch.cfg
httpfetch.exe
httpfetchspider.exe
importslave.exe
omnislave.cfg
omnislave.exe
pdfslave.exe
phraselist.dat
phraselist doc.dat
phraselist ppt.dat
phraselist qxd.dat
startfetch.sh
stopfetch.sh
uninstall.sh
```

4. Enter the command **cd httpfetch-2.3.x** to move to this new directory.

5. Run the installer script , **./Setup.sh** with the following arguments:

**<Base\_Name>**

Enter a unique name for the HTTP Fetch installation. Note that the unique name must not contain any spaces.

**<Main dir (fully qualified)>**

Enter the full path of the main installation directory.

**<DiSH Service Port>**

Enter the port number that HTTP Fetch will use for DiSH communication.

**<User Agent Contact Details>**

Enter some contact information, such as an e-mail address, so that the site that HTTP Fetch spiders can contact you.

**<IP Address for DRE>**

Enter the IP address (or name) of the machine on which the DRE is running.

**<Index Port Number for DRE>**

Enter the port that HTTP Fetch uses to index information into the DRE.

**<Database name in DRE>**

Enter the name of the DRE database into which you want HTTP Fetch to index the documents that it fetches.

6. The settings that you have entered are displayed. Confirm the settings by pressing **Return**. Alternatively, cancel the installation by pressing **Ctrl+C**.
7. HTTP Fetch has been installed successfully. Click on **Finish** to exit the installation.

You can now edit the HTTP Fetch configuration file (which is located in the main installation directory) and start the HTTP Service.

## Directory structure: UNIX

Once the installation of HTTP Fetch is completed, your installation directory will contain the following files:

<BaseName>.cfg	Configuration file that contains the HTTP Fetch settings.
<BaseName>.exe	HTTP Fetch executable.
<BaseName>spider.exe	Executable for the HTTP Fetch spider.
Setup.sh	Setup script for HTTP Fetch.
startfetch.sh	Startup script for HTTP Fetch.
stopfetch.sh	Stop script for HTTP Fetch.
Various TXT files	Text conversion tables used during the importing process.
uninstall.sh	Executable to uninstall HTTP Fetch from your computer.
binslave.cfg	Configuration file for Binslave.
binslave.exe	Executable that parses binary files looking for ASCII content, which can be imported into the DRE.
importslave.exe	Executable that generates IDX files for the DRE.
omnislave.cfg	Omnislave configuration file that contains the Omnislave settings.
omnislave.exe	Omnislave executable that parses files not in HTML or PDF format to IDX files.
pdfslave.exe	Executable that parses PDF files to IDX files.
phraselist.dat	File that Binslave uses.
phraselist doc.dat	File that Binslave uses.
phraselist ppt.dat	File that Binslave uses.
phraselist qxd.dat	File that Binslave uses.

### 3. Getting Started

---

In this quick start section we will run through the process of adding a new site to the HTTPFetch configuration and making sure we are only retrieving the contents that we want.

#### Adding a new site

It is important to provide the HTTPFetch with as much information as possible about what you wish to retrieve for a number of reasons:

- To limit the amount of bandwidth required locally.
- To reduce the amount of time required for indexing of content (there is no point in re-indexing the same content over and over if it never changes ).
- To reduce the load on the site you're fetching content from. Owners of web sites may become irate if they find that there is an automated fetch program hitting their site continuously.

We will use the HTTPFetch Administration Utility to carry out this process. It is not available on UNIX platforms so if you are running UNIX, you will need to edit the HTTPFetch configuration file manually or use an administration Utility running on a windows platform and point it at the configuration file. To do this, you will need to have samba or similar installed. This allows you to see the file system of the UNIX machine from Windows.

The Fetch Admin should have been installed during the installation of the HTTPFetch itself. To start the configuration module simply navigate to the '**Configure the HTTPFetch**' shortcut which can be found through the **Start -> HTTPFetch** menu.

It is also possible to edit the HTTPFetch Configuration file through Autonomy's DiSH (Distributed Service Handler) if you have it.. For this, please refer to the Service Settings appendix at the end of this manual and the DiSH manual itself.

In this quick start we will create a new entry in the HTTPFetch configuration to retrieve news and press release documents from the Autonomy web site. We will try to ensure that we are only retrieving recent documents and only those relating to news rather than background information about the company etc.

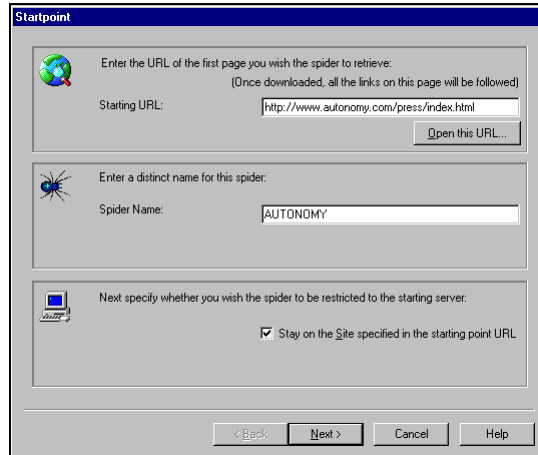
N.B. You will find that when you first use the Fetch Administration Utility, you already have an Autonomy spider set up. This example of how to add a new site to spider is, however, based around the Autonomy web site because of possible legal issues which could arise if another web site were to be used. Please note that this is based on the site at the time of going to press. Please be aware that this example should only be taken as a guide to how such a process is carried out.]

To start adding a new site simply click on the **New** button on the start page of the HTTPFetch Administration Utility. This will take you to the first page of the 'wizard' that we will use to enter the details of the site that we wish to add. This 'wizard' provides a method of selecting the directories that need spidering.

## Getting Started

### Startpoint

The first, and most important, step is to specify the starting point for our new site. At this stage all we need to do is to specify a name for our new site. This name must be a reasonable name like the one shown below and should definitely not be a URL. You will also need to specify the URL which the HTTPFetch should use as a starting point for its work.



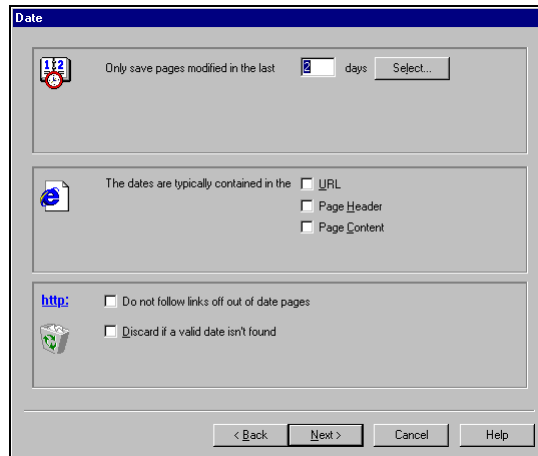
The **Startpoint** dialog box is used to configure the initial settings for the spider. It contains three sections:

- Enter the URL of the first page you wish the spider to retrieve:** (Once downloaded, all the links on this page will be followed)  
Starting URL:
- Enter a distinct name for this spider:**  
Spider Name:
- Next specify whether you wish the spider to be restricted to the starting server:**  
☒ Stay on the Site specified in the starting point URL

Navigation buttons at the bottom: , , , .

### Initial Criteria

You will then be asked details of the initial criteria with which the spider will run. This includes how old you wish the documents to be and where to look for dates in the documents.



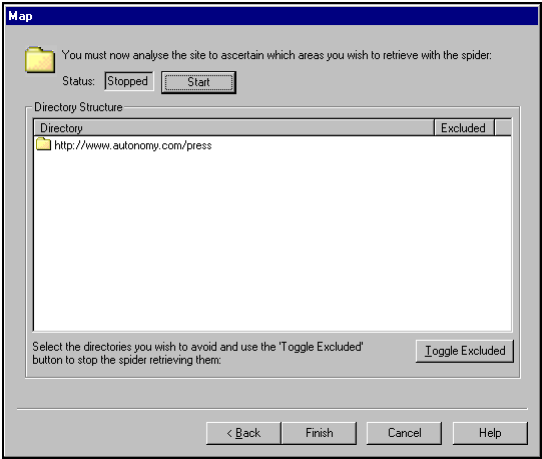
The **Date** dialog box is used to configure the criteria for the spider's run. It contains three sections:

- Only save pages modified in the last**  **days**
- The dates are typically contained in the**  
☐ URL  
☐ Page Header  
☐ Page Content
- http:**  
☐ Do not follow links off out of date pages  
☐ Discard if a valid date isn't found

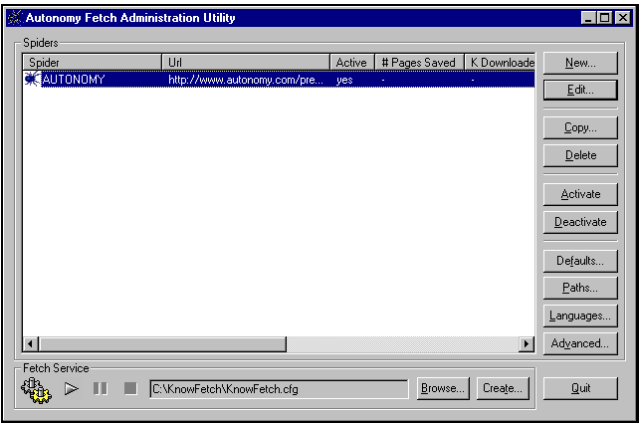
Navigation buttons at the bottom: , , , .

**Selecting directories to exclude from the spider**

The next screen will enable you to analyze the site from the start point and select which directories you do not wish to include in the spider. To exclude directories from the spider simply highlight the appropriate directory and click on the **Toggle Excluded** button at the bottom of the screen.



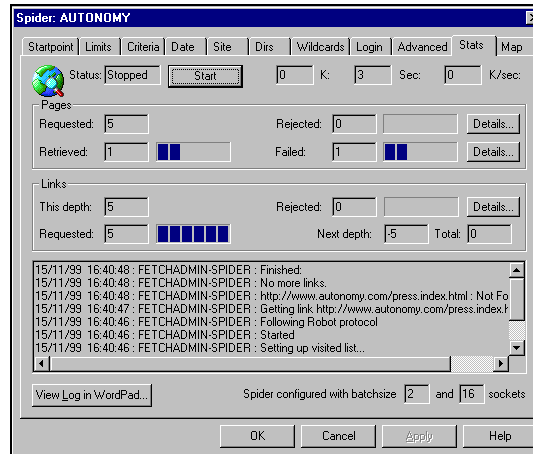
Once you have finished specifying the initial criteria for the spider, you will see the spider included in the list of spiders in the main HTTPFetch Administration Utility Window.



### Checking for a correct configuration

Having added our new site we are now ready to try it out. Rather than running the HTTPFetch we can test our configuration before we try it for real. Simply select the AUTONOMY entry on the main page of the HTTPFetch Administration Utility and then click on **Edit**. This will take you to a set of property pages for this site. Click the tab labeled **Statistics** on this screen. This will take you to a page that contains a number of blank fields concerning properties of the site.

If you click the **Start** button at the top of this page then the tool will carry out a test run of the fetching process to collect statistics about the configuration we've entered. This test process simply carries out an analysis of the web site specified, it doesn't actually save the pages which it encounters.



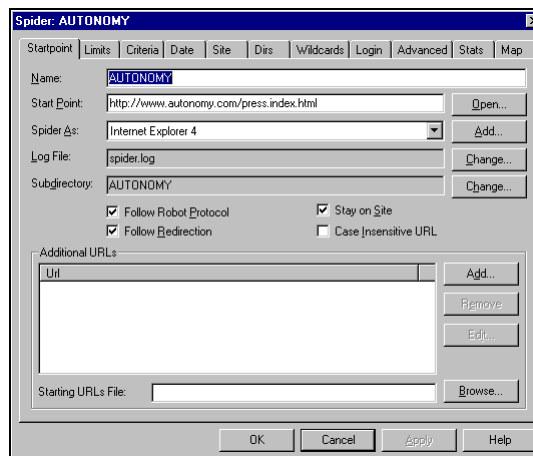
The screenshot shows the 'Spider: AUTONOMY' window with the 'Stats' tab selected. The window displays various statistics for the spider's operation. The 'Status' is 'Stopped'. The 'Start' button is visible. The 'Pages' section shows 'Requested: 5', 'Retrieved: 1', 'Rejected: 0', and 'Failed: 1'. The 'Links' section shows 'This depth: 5', 'Requested: 5', 'Next depth: -5', and 'Total: 0'. A log window at the bottom shows the following messages:

```
15/11/99 16:40:48: FETCHADMIN-SPIDER: Finished.  
15/11/99 16:40:48: FETCHADMIN-SPIDER: No more links.  
15/11/99 16:40:48: FETCHADMIN-SPIDER: http://www.autonomy.com/press.index.html: Not Fo  
15/11/99 16:40:47: FETCHADMIN-SPIDER: Getting link http://www.autonomy.com/press.index.f  
15/11/99 16:40:46: FETCHADMIN-SPIDER: Following Robot protocol  
15/11/99 16:40:46: FETCHADMIN-SPIDER: Started  
15/11/99 16:40:46: FETCHADMIN-SPIDER: Setting up visited list...
```

At the bottom, it says 'Spider configured with batchsize 2 and 16 sockets'.

### Editing the spider

Now we will edit the spider criteria. Highlight the appropriate spider and click on the **Edit** button. You will have displayed property sheet pages where you will find a large number of different options and parameters available, the majority of these are optional (or have default values) and for this example we will only be using and describing a subset of these.



The screenshot shows the 'Spider: AUTONOMY' window with the 'Criteria' tab selected. The window displays various configuration options for the spider. The 'Name' is 'AUTONOMY'. The 'Start Point' is 'http://www.autonomy.com/press.index.html'. The 'Spider As' is 'Internet Explorer 4'. The 'Log File' is 'spider.log'. The 'Subdirectory' is 'AUTONOMY'. The 'Follow Robot Protocol' checkbox is checked. The 'Follow Redirection' checkbox is checked. The 'Stay on Site' checkbox is checked. The 'Case Insensitive URL' checkbox is unchecked. The 'Additional URLs' section is empty. The 'Starting URLs File' is empty.



The **Name** used for a new site is purely to identify distinct spiders and is used by the fetch to identify directories, log files and other information associated with the spider. There are only two restrictions on this name. Firstly, the name must be unique for this installation of the HTTPFetch, i.e. there are no other entries in the configuration for this HTTPFetch which have the same name. Secondly, the name must contain only characters that are valid in filenames and directory names on the platform on which you are running. For this example let's simply call it AUTONOMY.

Secondly, we need to specify the starting point for the site. It might seem very obvious that the starting point for the site should simply be the site's front-page (<http://www.autonomy.com/>), however this is not always the case. In this example we stated that we wanted to retrieve only news and press documents. Most web sites structure the information that they are storing in a logical manner so as to make administration and navigation easier. The Autonomy web site is no different.

If we take a look at the web site itself we can see that the information we want is actually stored in a particular part of the web site. Following the navigation button *Press Office* on the front page directs us to a particular part of the web site ... <http://www.autonomy.com/press/index.html> ... and it is from here that we find all of the links to news / press documents which we want to examine. So, by taking this as our **Start Point**, we are already reducing the processing / communication overhead for this site.

Lastly, so as to be able to track what the HTTPFetch is actually doing for this site we can specify that it is to have its own log file. By default all the sites we add to the configuration for the fetch will append their activities to the file called **FetchName.log**. To change this simply click on the **Change** button next to the **Log File** entry and enter a new name for a file in the dialog box that appears.

These settings are the only ones we have to supply. All of the others parameters have default values that we can simply accept.

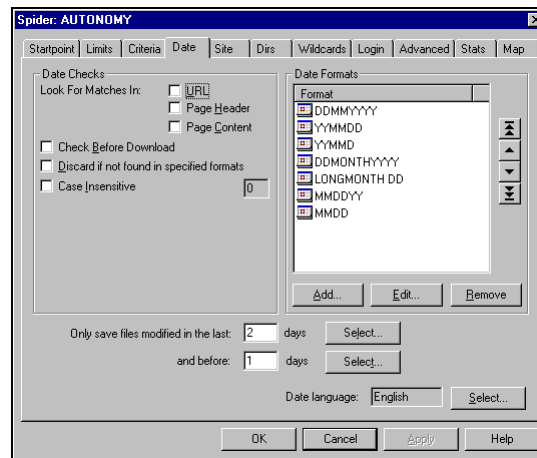
### Limits & Criteria

The Limits & Criteria property sheets let us specify the physical characteristics of retrieving pages from our site. The default values for both these steps are adequate for our needs and we can simply continue without making any additions or changes.

### Date

Giving the HTTPFetch information about how documents are dated is critical if we want to reduce duplication of work and only retrieve those documents that have actually changed since the last time we visited the site.

If we return to the web site we are spidering and follow one of the links to an actual article (e.g. [http://www.autonomy.com/press/pr\\_compaq.html](http://www.autonomy.com/press/pr_compaq.html)) we can see that the page contains a date for when the article was published (in this case it was February 17th 1999). We can instruct the HTTPFetch about the presence of this piece of information by adding the following settings in this property sheet. We need to add three pieces of information for correct processing of pages.



Firstly, we need to tell the HTTPFetch where the date is present. This can be in the **URL**, the **header** or the actual **content** of the HTML page. Find out where the date is stored and check the appropriate checkbox. The other options present tell the fetch how specific to be. Whether or not to download the page if the date is not in the right format, if it is to be case sensitive and whether to download or follow the links from a page if it does not contain a date at all (i.e. not saving the page).

Secondly, we need to tell the HTTPFetch what format the date appears in. To do this, add an entry by clicking on the **Add** format and enter the format the date appears in. If for example the date was **February 17, 2000** then the format to enter would be "**LONGMONTH DD, YYYY**". We need to include the quotes to make it explicit that a comma (or any other character) is included within the date format otherwise they are not needed. Please refer to the 'Supported Date Formats for Spidering' appendix for the available formats with descriptions and examples.

Lastly, we need to alter the **Only save files created after / before date** entries so that they are set to receive files within a certain date range. For example setting it to **1** and **1** will tell the HTTPFetch to save only those files that are identified as having been published between yesterday and tomorrow.

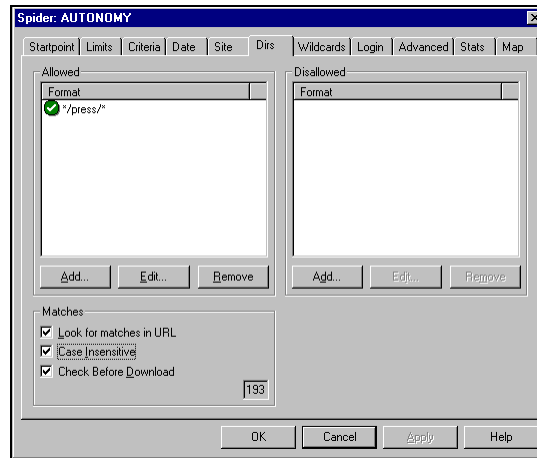
### Site & Dirs

Should we need to spider pages off site, then we can restrict which sites are actually to be spidered by specifying it in the Site Property Sheet. So for example if a site had several host names but had a common string in the names and we only wanted to spider this site then spider would have to be allowed to go off site (set in the initial setup of the spider). To get around this, the common string could be entered into the allowed box i.e. **\*news\***. This means **news** has to be found in the host name to continue with that site. Similarly this can be used to disallow host names.

As was stated above, web sites are in general structured in a logical manner so as to make administration and navigation easier. The Autonomy web site is no different. All of the pages that we are interested in are located within or below the <http://www.autonomy.com/press/> directory. We can use this information to our advantage to reduce processing overheads by including this in the wildcard property sheet.

Simply add **\*/press/\*** as a **NavDirsCSVs** entry in the Dir Property Sheet **Allowed**. Make sure that the **Look For Matches In URL** checkbox is checked (so as to look for matches in the URL). Also the **Check Before Download** checkbox should be checked (to make sure no pages without **/press/** in the directory are downloaded nor links followed off them).

The affect of this is that the HTTPFetch will check each page and only save those pages whose URL that contain the string **\*/press/\***, i.e. everything in or below this publishing directory on the web site.



### Wildcards

Again, we could restrict the pages that are downloaded according to strings that appear in the URL, header or content of the page. For our example, this is not necessary so again we will leave it blank.

### Login & Advanced

The last property sheets that we need to look at let us specify advanced features for this site. They enable us to use cookies or passwords to gain access to restricted sites as well as make use of special protocols and security for communication with the web site from which we are retrieving pages. We can accept the values given to us and click **OK** to complete the process of adding a new site to our HTTPFetch configuration.

Once you have edited the spider, test run the spider again. Once this test process has finished we can examine the statistics for the site. In the **Pages** section of the page you should be able to see how many HTML pages have been retrieved. Initially you're likely to see that all of the retrieved pages have been rejected. You can find out why this was by clicking the **Details** button next to the **Rejected** entry.

You'll probably see that everything has been rejected because of the **Date Range**. This is because we stated that we only want to save pages that were created with the last day or less. As a test go back to the property page where we specified the date checking to use and change the **Only save files created after date** to be something very far in the past (e.g. -999). Then run the statistics process again and see the difference this has made. You should see that at least some pages have now been saved successfully because of our changes.

## Running the process

Simply quit the HTTPFetch Administration Utility and (re)start the HTTPFetch to (re)run the process but to include our new site.

When the HTTPFetch is run (by default) it creates a sub-directory for each site from which it is retrieving documents. While the process runs you should be able to see a number of files being created and amended. The types of files are listed below:

- **Allurls**

During a single cycle of the fetch this file will build up a list of all of the URLs which have already been encountered so as not to duplicate effort.

- **Depth**

The structure of an Internet web site can be regarded as having a hierarchical structure which stores information at different depths. The page that is returned from the Start Point URL is regarded as being stored at depth 0. The pages that are returned from the links found in files at depth 0 are regarded as being at depth 1 and so on. As the fetch runs the typical pattern to see is that the deeper the process goes the more files it encounters (i.e. the depth files become larger at greater depths).

- **HTML**

Any pages, which are successfully saved from the web site, are temporarily saved in the sub-directory as HTML files so that they can be processed further later.

- **Log**

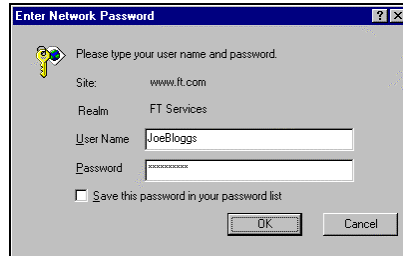
If you have specified a particular log file for this site then it will also be created in the sub-directory. If you haven't specified a log file name then the default behavior is to append activities for all sites to a single log file called spider.log which is created in the base directory of the HTTPFetch. **THIS SHOULD BE THE FIRST PLACE TO LOOK FOR ERRORS IF YOU FEEL THAT THE FETCH IS NOT BEHAVING AS YOU EXPECT.**

## Using the Login settings for secure sites

Some web sites employ a level of security, which allows them to limit access to a privileged set of clients (a subscription web site) or to provide clients with more personalized information. You can configure the HTTPFetch to give your users access to such sites by providing it with the information that the respective site requires (for example, username, password etc.).

### For example:

A large section of *The Financial Times* web site (<http://www.ft.com/>), which provides financial business news, is protected by authentication. If you try to access the site you will be presented with an **authenticate** dialog, which prompts you for login details:



To fetch documents from this site you need to provide the HTTPFetch with the following information, using the **Edit Spider** dialog's **Login** page:

<b>Method</b>	The method of security authentication that the web site uses. The HTTPFetch supports two main types of authentication: <ul style="list-style-type: none"> <li>• <b>Authentication dialogs</b> (including cookie-based logins).</li> <li>• <b>Form based logins</b> where the login information is entered into the HTML page that is presented to the user.</li> </ul>
<b>URL</b>	The URL where login details are be submitted.
<b>Username</b>	The account user's name.
<b>Password</b>	The account user's password.

## Using NTLM Proxy to access sites that use NTLM authentication

If a site uses NTLM authentication you need to install the NTLM Proxy module in order to enable HTTPFetch to access this site. You can then configure HTTPFetch to point at the NTLM Proxy, so that it can feed back the site to the HTTPFetch (See appendix **The NTLM Proxy module**).

## 4. Configuring HTTPFetch

---

The settings that determine how HTTPFetch operates are contained in the **<InstallationName>** configuration file, which is located in your installation directory. You can modify these settings in order to customize HTTPFetch according to your requirements.

### Entering Boolean values

For parameters that require Boolean settings the following settings are interchangeable

TRUE = true = ON = on = Y = y = 1

FALSE = false = OFF = off = N = n = 0

### Entering string values

If the value that you want to enter for a parameter that requires a string contains quotation marks, you must put the value into quotation marks and escape each quotation mark that the string contains by putting a slash in front of it.

#### For example:

```
FIELDSTART0="<font face=\"arial\"size=\"+1\"><b>"
```

Here the beginning and end of the string is indicated by quotation marks while all quotation marks that are contained in the string are escaped.

If you want to enter a comma separated list of strings for a parameter, and one of the strings contains a comma, you must indicate the start and the end of this string with quotation marks.

#### For example:

```
ParameterName=cat,dog,bird,"wing,beak",turtle
```

If any string within a comma separated list contains quotation marks, you must put this string into quotation marks and escaped the quotation marks in the string by putting a slash in front of them.

#### For example:

```
ParameterName="<font face=\"arial\"size=\"+1\"><b>","dog,bird,"wing,beak",turtle
```

### Applying modifications to HTTPFetch's operation

New configuration settings only take effect once the HTTPFetch service is stopped and restarted.

## Configuration file sections

The HTTPFetch configuration file contains the following sections:

[License]  
[Service]  
[Default]  
[Spider]  
[<MySpider>]

The number of sections depends on the number of [<MySpider>] sections specified in the configuration file.

**Note:**

- You should take care not to overwhelm your Internet connection by using up all available bandwidth, or by requesting too much information from a given site in too short a time. This can prevent other users from accessing the site and may cause your IP address to be banned from this site. This chapter and the Tutorial chapter will instruct you on how to prevent this from happening.
- For import parameters that you can specify in the configuration file's [Default] and [<MySpider>] sections, see the Autonomy Import Module manual.

### [License] section

This section contains the licensing details. You should not edit this section, as this could stop HTTPFetch working.

**Holder**

The name of the license holder.

**Key**

The license key.



**[Service] section**

If you use HTTPFetch in conjunction with DiSH, this section determines which machines are permitted to use and control the DiSH service.

**ServicePort**

Specify the port that the Fetch uses for DiSH communication.

**ServiceControlClients**

Specify the IP address of machines that are permitted to control HTTPFetch.

If you want to permit a number of machines to control HTTPFetch, you must separate the individual addresses with a comma. For example, **196.172.86.220,196.172.87.11** (note that there is no space before or after the comma).

Alternatively, you can use a wildcard in the IP address. Enter for example **187.\*.\*** to permit any machine whose IP address begins with 187 to control HTTPFetch.

**ServiceStatusClients**

Specify the IP address of machines that are permitted to access information about the HTTPFetch status but are not permitted to control it.

If you want to permit a number of machines to access the HTTPFetch status, you must separate the individual addresses with a comma. For example, **196.172.86.220,196.172.87.11** (note that there is no space before or after the comma).

Alternatively, you can use a wildcard in the IP address. Enter for example **187.\*.\*** to permit any machine whose IP address begins with 187 to access the HTTPFetch status.

## [Default] section

HTTPFetch uses the values that you set in the **[Default]** section for all jobs that you define in **[<MySpider>]** sections by default. If you want to specify different settings for an individual spider, you can set them in the **[<MySpider>]** section, in which case the default settings for this job will be overridden.

### DREHost

Specify the IP address (or name) of the machine on which the DRE is running (that is the DRE into which you want HTTPFetch to index the documents it fetches).

For example:

**DREHost=127.0.0.1**

### IndexPort

Specify the port number that HTTPFetch uses to index information into the DRE.

For example:

**IndexPort=8001**

### Database

Specify the name of the DRE database into which you want HTTPFetch to index the documents that it fetches.

For example:

**Database=News**

### LogFile

Specify the name of the log file that is generated for HTTPFetch. By default this is **spider.log**.

### LogFileMaxSize

Specify the maximum file size in kilobytes that a log file can reach before it is backed up and a new one is generated. The default setting is **4096**.

**LoginFieldName**

If you must enter more information than a username and password when logging on to a site that you want to spider, you can use **LoginFieldName** to specify the name of a field tag that takes an additional value. You use **LoginFieldValue** to specify the value that you enter in this field. **Note:** you must number each pairing of these parameters and the numbering must start from **0**.

For example:

**LoginFieldName0=Email**

**LoginFieldValue0=user@mycompany.com**

In this example, the site requires you to enter the e-mail address **user@mycompany.com** in the field called **Email**.

**LoginFieldValue**

Specify the value that you enter in the field specified for **LoginFieldName** when you log on to the site that you want to spider. **Note:** you must number each pairing of these parameters and the numbering must start from **0**.

For example:

**LoginFieldName0=E-mail**

**LoginFieldValue0=user@mycompany.com**

In this example, the site requires you to enter the e-mail address **user@mycompany.com** in the field called **Email**.

**LoginMethod**

Specify the method to use when logging on to the site that you want to spider. You can enter one of the following:

**AUTHENTICATE**

Enter this if you enter details (such as user name and password) into a dialog box that appears when you log onto a site.

**FORM**

Enter this if you enter details (such as user name and password) into fields on the site and the form uses the method **get** to send these details to the site's server. **Note:** view the source code of the page to determine the method that the form uses.

**FORMPOST**

Enter this if you enter details (such as user name and password) into fields on the site and the form uses the method **post** to send these details to the site's server. **Note:** view the source code of the page to determine the method that the form uses.

### LoginPassField

Specify the name of the field tag that takes the user password when you log on to the site that you want to spider. You use **LoginPassValue** to specify the password that you enter.

For example:

**LoginPassField=Password**

**LoginPassValue=me201**

In this example, the site requires you to enter the password **me201** in the field called **Password**.

### LoginPassValue

Specify the password that you enter in the field specified for **LoginPassField** when you log on to the site that you want to spider.

For example:

**LoginPassField=Password**

**LoginPassValue=me201**

In this example, the site requires you to enter the password **me201** in the field called **Password**.

### LoginURL

Specify the URL where a user logs on to the site that you want to spider.

For example:

**LoginURL=http://domain.server.com/login.html**

### LoginUserField

Specify the name of the field tag that takes the user name when you log on to the site that is being spidered. You use **LoginUserValue** to specify the user name that you enter.

For example:

**LoginUserField=Username**

**LoginUserValue=MyUser**

In this example, the site requires you to enter the user name **MyUser** in the field called **Username**.

### **LoginUserValue**

Specify the user name that you enter in the field specified for **LoginUserField** when you log on to the site that you want to spider.

For example:

**LoginUserField=Username**

**LoginUserValue=MyUser**

In this example, the site requires you to enter the user name **MyUser** in the field called **Username**.

### **MaxGlobalKBPS**

Specify the maximum bandwidth in kilobits per second that all of the spiders may use at any one time. By default this is **8192**.

### **MaxKBPS**

Specify the maximum bandwidth in kilobits per second that any individual spider may use at any one time. By default this is **4096**.

### **NSockets**

Specify the number of sockets to use in parallel for downloading data from a repository. By default this is **16**.

### **SpiderCycles**

Specify the number of cycles that you want the Fetch to perform. If you want the Fetch to cycle indefinitely, enter **-1** (this is the default setting).

### **SpiderStartTime**

Specify the time when the Fetch will start. Use the 24 hour clock when you specify the start time (hh:mm). By default this is the **now**.

### SpiderRepeatSecs

Specify the amount of time that elapses between each Fetch cycle. You can enter one of the following:

#### **n seconds**

Enter the number of seconds that you want to elapse between Fetch cycles. For example, if you enter **86400 seconds** (the default value), the Fetch cycle will run every 24 hours (24 hours=86400 seconds).

#### **n minutes**

Enter the number of minutes that you want to elapse between Fetch cycles followed by **minutes**. For example, if you enter **10 minutes**, the Fetch cycle will run every 10 minutes.

#### **n hours**

Enter the number of hours that you want to elapse between Fetch operations followed by **hours**. For example, if you enter **1 hour**, the Fetch cycle will run every hour.

#### **n days**

Enter the number of days that you want to elapse between Fetch cycles followed by **days**. For example, if you enter **1 day**, the Fetch cycle will run every day.

#### **n weeks**

Enter the number of weeks that you want to elapse between Fetch cycles followed by **weeks**. For example, if you enter **1 week**, the Fetch cycle will run every week.

#### **n months**

Enter the number of months that you want to elapse between Fetch cycles followed by **months**. For example, if you enter **1 month**, the Fetch cycle will run every month.

**Note:** If you enter a number without specifying a qualifier, the Fetch automatically treats the entry as seconds.

### Depth

Specify the maximum depth that the Fetch is allowed to go down to when spidering. By default this is **3**.

For example:

#### **Depth=10**

From the first page that the Fetch retrieves it can follow all links that this page contains to the next level (that is the first level), then follow all links from there to the next level (that is the second level) and then follow all links from there to the next level (that is the third level). Having reached this level the Fetch cannot follow any more links.

### **SiteDuration**

Specify the maximum amount of time in seconds that a spider is allowed to spend on one site. By default this is **43200** (12 hours).

### **MaxPages**

Specify the maximum number of pages that a spider is allowed to download per fetch cycle. By default this is **2000**.

### **MaxRetries**

Specify the maximum number of times that the Fetch can try to download a page when the first attempt fails. By default this is **3**.

### **FollowRedirect**

Enter **true** if you want the spider to follow HTTP redirects. This is the default.

Enter **false** if you do not want the spider to follow HTTP redirects.

### **StayOnSite**

Enter **true** if you do not want the spider to follow links that lead away from the original URL website that the Fetch spiders. This is the default.

Enter **false** if you want the spider to follow links that lead away from the original URL website.

### **MinPageSize**

Specify the minimum size (in bytes) that a page must be for the Fetch to download it. By default this is **4090**.

### **MaxPageSize**

Specify the maximum size (in bytes) that a page can be for the Fetch to download it. By default this is **163840**.

### **MaxLinksPerPage**

Specify the maximum number of links that a page can have for the Fetch to download it. By default this is **100**.

### **PageTimeOut**

Specify the maximum number of seconds that the Fetch will wait to download a page. If it does not manage to download data within this time span, it times out. By default this is **100**.

### **CookieName**

If you need to present a cookie when logging on to the site that you want to spider, you use **CookieName** to specify the name of the cookie. You use **CookieValue** to specify the value of the cookie.

For example:

**CookieName0=MyCookie**

**CookieValue0=01010101010101**

In this example, the cookie **MyCookie** with the value **01010101010101** is presented when the Fetch logs onto the site.

### **CookieValue**

If you need to present a cookie when logging on to the site that you want to spider, you use **CookieValue** to specify the value of the cookie specified for **CookieName**.

For example:

**CookieName0=MyCookie**

**CookieValue0=01010101010101**

In this example, the cookie **MyCookie** with the value **01010101010101** is presented when the Fetch logs onto the site.

### **CantHaveCheck**

Allows you to specify where and how the strings specified for **CantHaveCSVs** should be applied by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**URL: 1**

If you enter **1** the Fetch checks if the URL of a page contains any of the strings specified for **CantHaveCSVs**. If the URL does, the Fetch discards the page.

**Page header: 4**

If you enter **4** the Fetch checks if the header of a page contains any of the strings specified for **CantHaveCSVs**. If the header does, the Fetch discards the page.

**Page content: 8**

If you enter **8** the Fetch checks if the content of a page contains any of the strings specified for **CantHaveCSVs**. If the content does, the Fetch discards the page.



**Case insensitive: 64**

If you add **64** to the sum of one or more of the values specified above, the Fetch checks the appropriate page parts for any of the strings specified for **CantHaveCSVs**, and discards a page if it finds a case-insensitive match of any of the **CantHaveCSVs** strings.

**Note:** you cannot specify **64** on its own for **CantHaveCheck** as the Fetch would not know which parts of a page it should check for the **CantHaveCSVs** strings.

**Before download: 128**

If you enter **128**, the Fetch will check if a page contains any of the strings specified for **CantHaveCSVs** before it downloads it. If a page does contain any of the **CantHaveCSVs** strings, it is not downloaded.

**Note:** you cannot specify **128** on its own for **CantHaveCheck**.

**Valid site structure: 512**

If you enter **512**, the Fetch will recheck the **CantHaveCSVs** values for the site in order to make sure that the site is still valid before it updates it (if you don't make this setting, then changes to these values are never checked for). If the site is not valid, it is not downloaded.

For example:

**CantHaveCheck=5**

In this example the Fetch checks the URLs and the headers of pages for **CantHaveCSVs**. If the URL and/or the header of a page contains one of the **CantHaveCSVs** strings, the Fetch discards the page.

**CantHaveCheck=77**

In this example, the Fetch checks the URLs, the headers and the content of pages for case-insensitive matches of any **CantHaveCSVs**. If the URL, the header and/or the content of a page contain one of the **CantHaveCSVs** strings, the Fetch discards the page.

**CantHaveCSVs**

Specify one or more strings that a page may not contain in order to be used. Use **CantHaveCheck** to specify which page parts may not contain the specified strings and if pages should be checked for the specified strings before or after they are downloaded.

**Note:** if you want to specify multiple strings you must separate them with commas (with no spaces before or after a comma). You can use wildcards in the strings that you specify.

For example:

**CantHaveCheck=9****CantHaveCSVs=\*archive\*,\*test\***

In this example the Fetch checks if web pages contain the strings **archive** and **test** (as part of a word or whole word) in their URL or content. If this is the case, the page is discarded.

### **MustHaveCheck**

Allows you to specify where and how the strings specified for **MustHaveCSVs** should be applied by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**URL:**                    **1**

If you enter **1**, the Fetch checks if the URL of a page contains any of the strings specified for **MustHaveCSVs**. If the URL does not, the Fetch discards the page.

**Page header:**        **4**

If you enter **4**, the Fetch checks if the header of a page contains any of the strings specified for **MustHaveCSVs**. If the header does not, the Fetch discards the page.

**Page content:**       **8**

If you enter **8**, the Fetch checks if the content of a page contains any of the strings specified for **MustHaveCSVs**. If the content does not, the Fetch discards the page.

**Case insensitive:** **64**

If you add **64** to the sum of one or more of the values specified above, the Fetch checks the appropriate page parts for any of the strings specified for **MustHaveCSVs**, and discards a page if it does not find a case-insensitive match of any of the **MustHaveCSVs** strings.

**Note:** you cannot specify **64** on its own for **MustHaveCheck** as the Fetch would not know which parts of a page it should check for the **MustHaveCSVs** strings.

**Before download:** **128**

If you add **128** to the **URL** value, the Fetch will check if a page contains any of the strings specified for **MustHaveCSVs** before it downloads it. If a page does not contain the **MustHaveCSVs** strings, it is not downloaded.

**Note:** you cannot specify **128** on its own for **NavDirCheck**.

**Valid site structure:** **512**

If you enter **512**, the Fetch will recheck the **MustHaveCSVs** values for the site in order to make sure that the site is still valid before it updates it (if you don't make this setting, then changes to these values are never checked for). If the site is not valid, it is not downloaded.

For example:

**MustHaveCheck=5**

In this example the Fetch checks the URLs and the headers of pages for **MustHaveCSVs**. If the URL and/or the header of a page does not contain any of the **MustHaveCSVs** strings, the Fetch discards the page.

**MustHaveCheck=77**

In this example the Fetch checks the URLs, the headers and the content of pages for case-insensitive matches of any **MustHaveCSVs**. If the URL, the header and/or the content of a page do not contain any of the **MustHaveCSVs** strings, the Fetch discards the page.

### **MustHaveCSVs**

Specify one or more strings that a page must contain in order to be used. You use **MustHaveCheck** to specify which page parts must contain the specified strings and if pages should be checked for the specified strings before or after they are downloaded.

**Note:** if you want to specify multiple strings you must separate them with commas (with no spaces before or after a comma). You can use wildcards in the strings that you specify.

For example:

**MustHaveCheck=9**

**MustHaveCSVs=\*news\*,\*latest\***

In this example, the Fetch checks if the URL or content of a web page contains the strings **news** or **latest** (as part of a word or whole word). If this is not the case, the page is discarded.

### **BeforeDate**

You use **BeforeDate** and **AfterDate** to define the time span within which documents must have been modified for the Fetch to process them.

For example:

**BeforeDate=-3**

**AfterDate=4**

In this example, the Fetch only processes documents whose modification date lies between the current date **minus 3** days and the current day **plus 4** days (which, for example, can be the case if a document stems from another time zone). By default, **BeforeDate** has the value **-9000**.

### AfterDate

You use **BeforeDate** and **AfterDate** to define the time span within which documents must have been modified for the Fetch to process them.

For example:

**BeforeDate=-3**

**AfterDate=4**

In this example, the Fetch only processes documents whose modification date lies between the current date **minus 3** days and the current day **plus 4** days (which, for example, can be the case if a document stems from another time zone). By default, **AfterDate** has the value **-9999**.

### SpiderAs

Specify the text that is sent as the User-Agent details to the Web site that you are spidering.

By default this is **Mozilla/4.0 (compatible; MSIE 4.0: Win 32)**.

**<InstallationName>**

The installation name that you entered during the installation process.

**<ContactDetails>**

The contact details that you entered during the installation process.

### DateCheck

Allows you to specify where and how the Fetch should check for dates in the formats specified for **DateFormats** by entering a bitwise mask number. By default this is **0**, that is, the Fetch checks the date in the page's HTTP header. You can make up the number by adding up some of the following numbers as appropriate:

**URL: 1**

If you enter **1** the Fetch checks if the URL of a page contains any of the specified **DateFormats**. If the URL does not, the Fetch indexes the page using the date specified for **DefaultDate**. If no date has been specified for **DefaultDate**, the Fetch indexes the page using today's date.

**Page header: 4**

If you enter **4** the Fetch checks if the header of a page contains any of the specified **DateFormats**. If the header does not, the Fetch indexes the page using the date specified for **DefaultDate**. If no date has been specified for **DefaultDate**, the Fetch indexes the page using today's date.

**Page content: 8**

If you enter **8** the Fetch checks if the content of a page contains any of the specified **DateFormats**. If the content does not, the Fetch indexes the page using the date specified for **DefaultDate**. If no date has been specified for **DefaultDate**, the Fetch indexes the page using today's date.

**Case insensitive: 64**

If you add **64** to the sum of one or more of the values specified above, the Fetch checks the appropriate page parts for any of the specified **DateFormats**. If it cannot find a case-insensitive match of any of the **DateFormats**, the Fetch indexes the page using the date specified for **DefaultDate**. If no date has been specified for **DefaultDate**, the Fetch indexes the page using today's date.

**Note:** you cannot specify **64** on its own for **DateCheck** as the Fetch would not know which parts of a page it should check for the **DateFormats**.

**Discard if not found in specified formats: 256**

If you add **256** to the sum of one or more of the URL, Page header and Page content values, the Fetch checks the appropriate page parts for any of the specified **DateFormats**, and discards any page that does not contain one of the specified formats.

**Note:** you cannot specify **256** on its own but must add it to one or more of the values that specify a page part (that is the URL, Page header and Page content value). You can then add any of the other values (that is Case insensitive and/or Before download) as well.

For example:

**DateCheck=5**

In this example the Fetch checks the URLs and the headers of pages for **DateFormats**. If neither the URL nor the header of a page contain any of the specified **DateFormats**, the Fetch indexes the page using the date specified for **DefaultDate**.

**DateCheck=393**

In this example the Fetch checks the URLs and the content of pages for **DateFormats** before downloading the pages. If neither the URL nor the content of a page contain any of the specified **DateFormats**, the Fetch discards the page.

**DateLongMonthCSVs**

Enter one or more strings for which the Fetch checks when the format **LONGMONTH** has been specified for **DateFormats**. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). By default this is the twelve English months (that is, January, February, and so on).

For example:

**DateLongMonthCSVs=Januar,Februar**

**DateFormats=LONGMONTH/YYYY**

In this example, the Fetch checks for dates containing the strings **Januar** or **Februar** followed by a year in a four-digit format (for example, Januar/2002) when extracting dates.

### DateMonthCSVs

Enter one or more strings for which the Fetch checks when the format **SHORTMONTH** has been specified for **DateFormats**. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). By default this is the twelve standard abbreviations for English months (that is, Jan, Feb, and so on).

For example:

**DateMonthCSVs=Jan,Feb,Mar**

**DateFormats=SHORTMONTH/YYYY**

In this example, the Fetch checks for dates containing the strings **Jan**, **Feb** or **Mar** followed by a year in a four-digit format (for example, Jan/2002) when extracting dates.

### DateFormats

Enter one or more strings to specify which format the date can have that will be extracted from a web page. Use **DateCheck** to specify which page parts the Fetch should check for dates and whether the Fetch should discard a page if it does not contain any dates in the specified formats.

You can use the following format specifiers:

<b>YY</b>	Year (2 digit), for example, 99, 00, 01 and so on.
<b>YYYY</b>	Year (4 digit), for example, 1999, 2000, 2001 and so on.
<b>LONGMONTH</b>	For example January, March, August and so on.
<b>SHORTMONTH</b>	For example Jan, Mar, Aug and so on.
<b>MM</b>	Month (2 digit), for example, 01, 10, 12 and so on.
<b>M+</b>	Month (1/2 digit), for example, 1,2,3,10 and so on.
<b>DD</b>	Day (2 digit), for example, 01, 02, 03, 12, 23 and so on.
<b>D+</b>	Day (1 or more digits), for example, 1, 2, 12, 13, 31 and so on.
<b>LONGDAY</b>	(2 digit with postfix), for example, 1 <sup>st</sup> , 2 <sup>nd</sup> and so on.
<b>HH</b>	Hour (2 digit), for example, 01, 12, 13 and so on.
<b>H+</b>	Hour (1/2 digit)
<b>NN</b>	Minute (2 digit)
<b>N+</b>	Minute (1/2 digit)
<b>SS</b>	Second (2-digit)
<b>S+</b>	Second (1/2 digit)
<b>ZZZ</b>	Time Zone (for example, GMT, EST, PST, and so on.)

**Note:** If you want to specify multiple formats, you must separate them with commas (there must be no space before or after a comma). If you want to specify a format for a date that contains a space, you must put the format in quotation marks.

For example:

**DateFormats=D+/SHORTMONTH/YYYY, DDMMYY**

In this example only dates that have the format D+/SHORTMONTH/YYYY (for example, 2/Jan/2001) or DDMMYY (for example, 020101) can be extracted.

**DateFormats="D+SHORTMONTH YYYY", "Date: D+ LONGMONTH, YYYY"**

In this example only dates that have the format D+ SHORTMONTH YYYY (for example, 2 Jan 2001) or Date: D+ LONGMONTH, YYYY (for example, Date: 2 January, 2001) can be extracted.

### DatePostfixCSVs

Enter one or more strings that the Fetch recognizes as postfixes when the format **LONGDAY** has been specified for **DateFormats**. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). By default this is **st,nd,rd,th**.

### DefaultDate

Enter one of the following to specify which date that the Fetch will display for pages that it downloaded without a date:

#### NOW

The Fetch displays the date when the document was downloaded. This is the default.

#### yyyy/mm/dd

The Fetch displays the specified year, month and day. If you change the format of the date (for example to dd/mm/yyyy), you must reflect this change in the DRE configuration file's date format to ensure that the date fields are consistent.

#### -ndays

The Fetch subtracts the specified *n* number of days from the current date (that is the date when the document was downloaded), and displays the resulting date.

#### +ndays

The Fetch adds the specified *n* number of days to the current date (that is the date when the document was downloaded), and displays the resulting date.

#### -nweeks

The Fetch subtracts the specified *n* number of weeks from the current date (that is the date when the document was downloaded), and displays the resulting date.

#### +nweeks

The Fetch adds the specified *n* number of weeks to the current date (that is the date when the document was downloaded), and displays the resulting date.

**-*n*years**

The Fetch subtracts the specified *n* number of years from the current date (that is the date when the document was downloaded), and displays the resulting date.

**+*n*years**

The Fetch adds the specified *n* number of years to the current date (that is the date when the document was downloaded), and displays the resulting date.

**DeleteStringCheck**

Allows you to specify where and how the Fetch should check for strings specified for **DeleteStringCSVs** by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**URL:**                   **1**

If you enter **1**, the Fetch checks if the URL of a page contains any of the specified strings. If the URL does contain the strings, the page is deleted from the DRE.

**Page header:**       **4**

If you enter **4**, the Fetch checks if the header of a page contains any of the specified strings. If the header does contain the strings, the page is deleted from the DRE.

**Page content:**       **8**

If you enter **8**, the Fetch checks if the content of a page contains any of the specified strings. If the content does contain the strings, the page is deleted from the DRE.

**Case insensitive:** **64**

If you add **64** to the sum of one or more of the values specified above, the Fetch checks the appropriate page parts for any of the specified strings. If it finds a case-insensitive match of any of the strings, the page is deleted from the DRE.

**Note:** you cannot specify **64** on its own for **DeleteStringCheck** as the Fetch would not know which parts of a page it should check for the strings specified for **DeleteStringCSVs**.

For example:

**DeleteStringCheck=5**

In this example, the Fetch checks the URLs and the headers of pages for strings specified for **DeleteStringCSVs**. If either the URL or the header of a page contain any of the specified strings, the page is deleted from the DRE.

**DeleteStringCheck=73**

In this example, the Fetch checks the URLs and the content of pages for case-insensitive matches to the strings specified for **DeleteStringCSVs**. If either the URL or the content of a page contain any of the specified strings, the page is deleted from the DRE.



**DeleteStringCSVs**

Enter one or more strings for which the Fetch checks and if it finds one of them, it marks the page as deleted and the DRE is updated accordingly. Use **DeleteStringCheck** to specify which page parts the Fetch should check for these strings and if pages should be checked for these strings before or after they are downloaded.

**Note:** if you want to specify multiple strings you must separate them with commas (with no spaces before or after a comma). You can use wildcards in the strings that you specify. For example:

**DeleteStringCSVs=\*archive\*,\*default\***

**DeleteStringCheck=5**

In this example, the Fetch checks the URLs and the headers of pages for the strings **archive** and **default** (as part of a word or whole word). If either the URL or the header of a page contain either of the strings, the page is deleted from the DRE.

**DeleteByField**

Enter the name of a referenceable field in the configuration file that contains the unique ID shared by a note and its attachments. HTTPFetch uses this field when deleting a document and its attachments from the DRE to make the deletion process more efficient.

**BatchProcess**

Enter **Import** if you want the Fetch to import files into an IDX file after the number of files specified for **Batch number** has been processed. Files are then indexed into the DRE once all files have been processed. This is the default.

Enter **Index** if you want the Fetch to index files into the DRE once the number of files specified for **Batch number** has been processed.

**BatchSize**

Specify the number of files that the Fetch processes before indexing or importing files as specified in **BatchProcess**. By default this is **200**.

**BatchStartDeleteOld**

Enter **true** if you want to delete any existing IDX files each time that you start HTTPFetch.

Enter **false** if you do not want to delete existing IDX files each time that you start HTTPFetch. This is the default.

**FollowRobotProtocol**

Enter **true** if you want the Fetch to follow robot protocol. This is the default.

Enter **false** if you don't want the fetch to follow robot protocol.

### **FollowRobotMeta**

If **FollowRobotProtocol** is set to **true**, you can use **FollowRobotMeta** to control whether the Fetch should follow the Robot metatags.

Enter **true** if you want the Fetch to follow the Robot metatags.

Enter **false** if the Fetch should not follow the Robot metatags. This is the default.

### **PageDelay**

Specify the number of seconds that elapses after the Fetch has finished downloading a page and before it requests the next page. By default this is **5**.

### **IndexMode**

Allows you to prevent the same document or document content being stored in your DRE more than once:

#### **REFERENCE**

If a document that is downloading has the same **DRREFERENCE** as a document that is already stored in the DRE, the document contained in the DRE is replaced with the new document. This is the default.

#### **REFERENCEMATCH<NN>**

If the content of a document that the Fetch is downloading is more similar than the <NN> percentage specified to the content of a document that is stored in the DRE, the document contained in the DRE is replaced with the new document.

For example, if you set **IndexMode** to **REFERENCEMATCH80**, and the Fetch downloads a document whose content is 80 % or more similar to a document that is already stored in the DRE, the document in the DRE is replaced by the new one.

#### **REFERENCE2MATCH<NN>**

If the content of a document that the Fetch is downloading is more similar than the <NN> percentage specified to the content of a document that is stored in any available DRE, the document contained in the available DRE databases is replaced with the new document. (Unlike **ReferenceMatch<NN>** which only replaces documents in the DRE into which the Fetch is indexing).

For example, if you set **IndexMode** to **Reference2Match80**, and the Fetch downloads a document whose content is 80 % or more similar to any document that is stored in any DRE databases, all instances of the document that are stored in any DRE database are replaced by the new one.

#### **NONE**

None of the documents that are stored in your DRE are replaced by new documents that the Fetch downloads.

**IndexOverSocket**

Enter **on** if the DRE into which the Fetch is indexing is installed on a different computer, so that the indexing of documents takes place over a socket. This is the default.

Enter **off** if the DRE and the Fetch are installed on the same computer.

**StoreSiteStructure**

Enter **true** if you want the Fetch to store site structures, so that the spidering process will be quicker if sites are spidered again. This is the default.

Enter **false** if you do not want the Fetch to store site structures.

**SiteStructureCompactCopy**

If **StoreSiteStructure** is set to **true**, **SiteStructureCompactCopy** allows you to specify whether you want to compact site structures.

Enter **true** if you want to compact site structures that are stored in the job directories.

Enter **false** if you do not want to compact site structures that are stored in the job directories.

**Extensions**

Allows you to restrict which document types the Fetch can spider by specifying one or more file extensions. If you want to specify multiple extensions, you must separate them with commas (there must be no space before or after a comma).

For example:

**Extensions=\*.html,\*.txt,\*.doc**

In this example HTTPFetch will only spider documents that have the extension **.html**, **.txt** or **.doc**.

**HTMLImportStartDefFlags**

Enter **true** to strip all metatags (apart from Autonomy metatags) from the header of downloaded pages. This is the default.

Enter **false** if you want to preserve all metatags that the header of downloaded pages contains.

**HTTPVersion**

Specify the HTTP protocol version, which is supported by the Web sites that you are spidering. The default version is **HTTP/1.0**.

### LinkCheck

Allows you to specify where the Fetch should check for strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs** by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**URL:**                    **1**

If you enter **1**, the Fetch checks if the URL of a page contains any of the strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs**.

**Header:**                **4**

If you enter **4**, the Fetch checks if the header of a page contains any of the strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs**.

**Page content:**        **8**

If you enter **8**, the Fetch checks if the content of a page contains any of the strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs**.

**Case insensitive:** **64**

If you add **64** to the sum of one or more of the values specified above, the Fetch checks the appropriate page parts for case-insensitive matches for the strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs**.

**Note:** you cannot specify **64** on its own for **LinkCheck** as the Fetch would not know which parts of a page it should check for the strings.

For example:

**LinkCheck=5**

In this example, the Fetch checks the URLs and headers of pages for case-sensitive matches of the strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs**.

**LinkCheck=73**

In this example, the Fetch checks the URLs and content of pages for case-insensitive matches of the strings specified for **LinkAllowCSVs** and **LinkDisallowCSVs**.

**LinkAllowCSVs**

Specify one or more strings that a page must contain for the Fetch to follow links from it. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). You can use wildcards in the strings. You use **LinkCheck** to specify which page parts the Fetch should check for these strings.

For example:

**LinkAllowCSVs=\*news\*,\*home\***

**LinkCheck=1**

In this example, if the Fetch finds a Web page with a URL that contains the strings **news** or **home** (as part of a word or a whole word), it processes this page and any page that links from it.

**LinkDisallowCSVs**

Specify one or more strings that a page must not contain for the Fetch to follow links from it. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). You can use wildcards in the strings. Use **LinkCheck** to specify which page parts the Fetch should check for these strings.

For example:

**LinkCheckDisallowCSVs=\*archive\*,\*test\***

**LinkCheck=1**

In this example, if the Fetch finds a Web page with a URL that contains the strings **archive** or **test** (as part of a word or a whole word), it processes this page but no pages that link from it.

**NavDirCheck**

Allows you to specify where the Fetch should check for strings specified for **NavDirAllowCSVs** and **NavDirDisallowCSVs** by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**URL: 1**

Enter **1** to specify that the Fetch checks if the URL of a page contains any of the strings specified for **NavDirAllowCSVs** and **NavDirDisallowCSVs**.

**Case insensitive: 64**

If you add **64** to the **URL** value, the Fetch checks the URL of a page for case-insensitive matches for the strings specified for **NavDirAllowCSVs** and **NavDirDisallowCSVs**.

**Note:** you cannot specify **64** on its own for **NavDirCheck**.

**Before download: 128**

If you add **128** to the **URL** value, the Fetch will check if a page contains any of the strings specified for **NavDirAllowCSVs** and **NavDirDisallowCSVs** before it downloads it.

**Note:** you cannot specify **128** on its own for **NavDirCheck**.

**Valid site structure: 512**

If you enter **512**, the Fetch will recheck the **NavDirAllowCSVs** and **NavDirDisallowCSVs** values for the site in order to make sure that the site is still valid before it updates it (if you don't make this setting, then the changes to these values are never checked for). If the site is not valid, it is not downloaded.

For example:

**NavDirCheck=65**

In this example, the Fetch checks the URLs of pages for case-insensitive matches of the strings specified for **NavDirAllowCSVs** and **NavDirDisallowCSVs**.

**NavDirCheck=193**

In this example, the Fetch checks the URLs of pages for case-insensitive matches of the strings specified for **NavDirAllowCSVs** and **NavDirDisallowCSVs**.

**NavDirAllowCSVs**

Enter one or more strings that the URL of a page must contain for the Fetch to process it. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). Use **NavDirCheck** to specify how and when the Fetch checks for these strings. You can use wildcards in the strings.

For example:

**NavDirAllowCSVs=\*news\*,\*home\***

In this example, a page must have a URL that contains the string news or home (as part of a word or a whole word) for the Fetch to process it.

**NavDirDisallowCSVs**

Enter one or more strings that the URL of a page must not contain for the Fetch to process it. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). Use **NavDirCheck** to specify how and when the Fetch checks for these strings. You can use wildcards in the strings.

For example:

**NavDirDisallowCSVs=\*archive\*,\*test\***

In this example, a page must not have a URL that contains the string **archive** or **test** (as part of a word or as a whole word) for the Fetch to process it.

**NavLinkDefEndCSVs**

Specify one or more strings that mark the end of custom links (for example, JavaScript links) that are not written using the standard HTML format for links. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). Use

**NavLinkStartCSVs** to specify strings that mark the start of custom links. **Note:** the strings correspond to the strings that you specify for **NavLinkDefStartCSVs**, **NavLinkDefPrefixCSVs** and **NavLinkDefPostfixCSVs**.

For example:

```
NavLinkDefStartCSVs=url:,"link:
NavLinkDefEndCSVs=>,"
NavLinkDefPostfixCSVs=/news/,/home/
NavLinkDefPrefixCSVs=.html,.htm
```

In this example, the Fetch recognizes any text displayed between **<url:** and **>** as a link and adds **/news/** to the start of the link text and **.html** to the end. The Fetch also recognizes any text displayed between **"link:** and **"** as a link and adds **/home/** to the start of the link text and **.htm** to the end.

**NavLinkDefStartCSVs**

Specify one or more strings that mark the start of custom links (for example, JavaScript links) that are not written using the standard HTML format for links. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). Use

**NavLinkEndCSVs** to specify strings that mark the end of custom links. **Note:** the strings correspond to the strings that you specify for **NavLinkDefEndCSVs**, **NavLinkDefPrefixCSVs** and **NavLinkDefPostfixCSVs**.

For example:

```
NavLinkDefStartCSVs=url:,"link:
NavLinkDefEndCSVs=>,"
NavLinkDefPostfixCSVs=/news/,/home/
NavLinkDefPrefixCSVs=.html,.htm
```

In this example, the Fetch recognizes any text displayed between **<url:** and **>** as a link and adds **/news/** to the start of the link text and **.html** to the end. The Fetch also recognizes any text displayed between **"link:** and **"** as a link and adds **/home/** to the start of the link text and **.htm** to the end.

### NavLinkDefPostfixCSVs

Specify one or more postfixes that the Fetch adds to custom links when extracting them. If you want to specify multiple postfixes, you must separate them with commas (there must be no space before or after a comma). **Note:** the postfixes correspond to the strings that you specify for **NavLinkDefStartCSVs**, **NavLinkDefEndCSVs** and **NavLinkDefPrefixCSVs**.

For example:

```
NavLinkDefStartCSVs=url:,"link:
NavLinkDefEndCSVs=>,"
NavLinkDefPostfixCSVs=/news/,/home/
NavLinkDefPrefixCSVs=.html,.htm
```

In this example, the Fetch recognizes any text displayed between **<url:** and **>** as a link and adds **/news/** to the start of the link text and **.html** to the end. The Fetch also recognizes any text displayed between **"link:** and **"** as a link and adds **/home/** to the start of the link text and **.htm** to the end.

### NavLinkDefPrefixCSVs

Specify one or more prefixes that the Fetch adds to custom links when extracting them. If you want to specify multiple prefixes, you must separate them with commas (there must be no space before or after a comma). **Note:** the postfixes correspond to the strings that you specify for **NavLinkDefStartCSVs**, **NavLinkDefEndCSVs** and **NavLinkDefPostfixCSVs**.

For example:

```
NavLinkDefStartCSVs=url:,"link:
NavLinkDefEndCSVs=>,"
NavLinkDefPostfixCSVs=/news/,/home/
NavLinkDefPrefixCSVs=.html,.htm
```

In this example, the Fetch recognizes any text displayed between **<url:** and **>** as a link and adds **/news/** to the start of the link text and **.html** to the end. The Fetch also recognizes any text displayed between **"link:** and **"** as a link and adds **/home/** to the start of the link text and **.htm** to the end.

### NavLinksToFollow

Allows you to specify what types of link the Fetch can follow by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**Follow all:**            **0X0FFFFFFF**

Enter **0X0FFFFFFF** to specify that the Fetch can follow all types of link.

**Frames:**                **1**

Enter **1** if you want the Fetch to follow **frame** links.



**Href:**                    **2**

Enter **2** if you want the Fetch to follow **href** links.

**Location:**            **4**

Enter **4** if you want the Fetch to follow **location.href** links.

**HTTP-equiv**           **8**

Enter **8** if you want the Fetch to follow **http-equiv** links.

For example:

**NavLinksToFollow=10**

In this example, the Fetch can follow **href** and **http-equiv** links but not **frame** and **location.href** links.

### **NavSiteAllowCSVs**

Enter one or more strings that the site name in the URL of a page must contain for the Fetch to process it. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). You use **NavSiteCheck** to specify how and when the Fetch checks for these strings. You can use wildcards in the strings.

**Note:** this can be useful if **StayOnSite** is set to **false** and you want to restrict the Fetch to processing pages from a site whose pages have URLs that contain a common site name.

For example:

**NavDirAllowCSVs=\*mycompany.com\***

In this example, a page must have a site name in the URL that contains the string **mycompany.com** for the Fetch to process it.

### **NavSiteCheck**

Allows you to specify how the Fetch should check for strings specified for **NavSiteAllowCSVs** and **NavSiteDisAllowCSVs** by entering a bitwise mask number. You can make up this number by adding up some of the following numbers as appropriate:

**URL:**                    **1**

Enter **1** to specify that the Fetch checks if the site name in the URL of a page contains any of the strings specified for **NavSiteAllowCSVs** and **NavSiteDisAllowCSVs**.

**Case insensitive: 64**

If you add **64** to the **URL** value, the Fetch checks the URL of a page for case-insensitive matches for the strings specified for **NavSiteAllowCSVs** and **NavSiteDisallowCSVs**.

**Note:** you cannot specify **64** on its own for **NavSiteCheck**.

**Before download: 128**

If you add **128** to the **URL** value, the Fetch will check if a page contains any of the **NavSiteDisallowCSVs** strings before it downloads it. If it does the Fetch does not download the page.

**Note:** you cannot specify **128** on its own for **NavSiteCheck**.

For example:

**NavSiteCheck=65**

In this example, the Fetch checks the URLs of pages for case-insensitive matches of the strings specified for **NavSiteAllowCSVs** and **NavSiteDisallowCSVs**.

**NavSiteCheck=193**

In this example, the Fetch checks the URLs of pages for case-insensitive matches of the strings specified for **NavSiteAllowCSVs** and **NavSiteDisallowCSVs**.

**NavSiteDisallowCSVs**

Enter one or more strings that the site name in the URL of a page must not contain for the Fetch to process it. If you want to specify multiple strings, you must separate them with commas (there must be no space before or after a comma). Use **NavSiteCheck** to specify how and when the Fetch checks for these strings. You can use wildcards in the strings.

For example:

**NavSiteDisallowCSVs=\*archive\*,\*test\***

In this example, a page must not have a site name that contains the string **archive** or **test** (as part of a word or as a whole word) for the Fetch to process it.

**ForceFirstPage**

Enter **true** if you want to always download the first page for a site, regardless of whether the web server reports any changes on the page (sometimes a web server reports no changes, even though links on the page have changed).

Enter **false** if you only want to download the first page for a site if the web server reports changes. This is the default.

**PageDuration**

Specify the length of time in seconds that the Fetch can attempt to download a page. By default this is **3600**.

**ProxyHost**

If you use a proxy server to access the Internet, specify the IP address of the proxy server that you use.

**ProxyPort**

If you use a proxy server to access the Internet, specify the port number on which the proxy server listens.

**ProxyUsername**

If you use a proxy server to access the Internet, specify your user name for the proxy server.

**ProxyPassword**

If you use a proxy server to access the Internet, specify your password for the proxy server.

**SpiderAccept**

Specify the string that the spider sends in its header to accept pages. By default this is **/\***. This means that it accepts all pages.

**SpiderRepeatInterval**

**Note:** this setting is overwritten by **SpiderRepeatSecs** if this setting is used.

The amount of time that elapses between each spider cycle. You can enter one of the following:

**n seconds**

Enter the number of seconds that you want to elapse between spider cycles. For example, if you enter **3600 seconds**, the spider cycle will run every hour (1 hour=3600 seconds).

**n minutes**

Enter the number of minutes that you want to elapse between spider cycles followed by **minutes**. For example, if you enter **10 minutes**, the spider cycle will run every 10 minutes.

**n hours**

Enter the number of hours that you want to elapse between spider operations followed by **hours**. For example, if you enter **1 hour**, the spider cycle will run every hour.

**n days**

Enter the number of days that you want to elapse between spider cycles followed by **days**. For example, if you enter **1 day**, the spider cycle will run every day (this is the default setting).

## Configuring HTTPFetch

### **n weeks**

Enter the number of weeks that you want to elapse between spider cycles followed by **weeks**. For example, if you enter **1 week**, the spider cycle will run every week.

### **n months**

Enter the number of months that you want to elapse between spider cycles followed by **months**. For example, if you enter **1 month**, the spider cycle will run every month.

**Note:** If you enter a number without specifying a qualifier, the Fetch automatically treats the entry as seconds.

### **SpiderShowReferer**

Enter **true** if you want the Fetch to enter a URL into the IDX file's **SpiderReferer** field, which points to the source page from which the fetched page linked.

Enter **false** if you do not want to be able to trace the source of linked pages. This is the default.

### **SpiderStartDeleteOld**

Enter **true** if you want the Fetch to delete temporary files from the previous cycle when a Fetch cycle starts.

Enter **false** if you do not want the Fetch to delete temporary files from the previous cycle when a Fetch cycle starts. This is the default.

### **URLCaseSensitive**

Enter **true** if you want URLs that are stored in a site structure to be case-sensitive. This is the default.

Enter **false** if you do not want URLs that are stored in a site structure to be case-sensitive.

### **URLFile**

Specify the full path to the file containing the list of URLs from which spidering starts.

### **URLFileDelete**

Enter **true** if you want the Fetch to delete the URL file after it has been read into the Fetch.

Enter **false** if you do not want the Fetch to delete the URL file after it has been read into the Fetch. This is the default.

### **MaxRequests**

Specify the maximum number of requests that a spider can make to a server during a fetch cycle. By default this is **2147483648**.

**SecurityType**

If you use SSL security, specify the type of security that you are using. Possible types are **SSL\_V23**, **SSL\_V2**, **SSL\_V3**, **TLS\_V1** and **SSL\_V23**.

**ClientCertificate**

If you use client-side SSL security, specify the full path to the client certificate (in PEM format) that is sent with requests.

For example:

**ClientCertificate=c:\secure\client.pem**

**SpiderPath**

Specify the full path to the location where you want to store spider files (e.g. structure files). By default this is the same location where the configuration file is located.

**URLn**

Specify the URLs where the spider starts. You must number each URL. The numbering starts from **0** and must be sequential.

For example:

**URL0=http://www.mycompany.com/**

**URL1=http://www.anothercompany.com/**

**TotalSize**

Specify the maximum number of bytes that a spider is permitted to download during one Fetch cycle. You can specify up to 4 billion. By default this is **1572864000**.

**TreatAsSlash**

Allows you to specify pages that you want to index as the root pages of sites. Enter one or more page names that the fetch strips from the URL of root pages of sites when indexing them into the DRE. If you want to enter multiple strings, you must separate them with a comma (there must be no space before or after a comma). By default this is **/INDEX.HTM,/INDEX.HTML,/HOME.HTM,/HOME.HTML,/DEFAULT.HTM,/DEFAULT.HTML,/INDEX.CGI**.

For example:

**TreatAsSlash=Default.htm,Index.htm**

In this example, the fetch strips the page names **Default.htm** and **Index.htm** from URLs and indexes these pages as the root pages of sites.

## Directory

Specify the full path to the directory where you want to store temporary files during the spidering process.

## SpiderRedirectReplaceCSVs

If a redirection that a spider is following is incorrect, **SpiderRedirectReplaceCSVs** allows you to correct the URL that re-directs the spider. Enter one or more string values that you want to replace with the corresponding **SpiderRedirectReplaceWithCSVs** strings in the URL (the first specified **SpiderRedirectReplaceCSVs** string with the first string specified for **SpiderRedirectReplaceWithCSVs**, the second string with the second string and so on). If you want to enter multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

**SpiderRedirectReplaceCSVs=/value1/,/value2/,/more\_values/**

**SpiderRedirectReplaceWithCSVs=/new\_value1/,/new\_value2/,/more\_new\_values/**

In this example, the URL's **/value1/** string is replaced with the **/new\_value1/** string, the **/value2/** string is replaced with the **/new\_value2/** string, and the **/more\_values/** string is replaced with the **/more\_new\_values/** string.

## SpiderRedirectReplaceWithCSVs

If a redirection that a spider is following is incorrect, **SpiderRedirectReplaceWithCSVs** allows you to correct the URL that re-directs the spider. Enter one or more string values with which you want to replace with the corresponding **SpiderRedirectReplaceCSVs** strings in the URL (the first specified **SpiderRedirectReplaceCSVs** string with the first string specified for **SpiderRedirectReplaceWithCSVs**, the second string with the second string and so on). If you want to enter multiple strings, you must separate them with commas (there must be no space before or after a comma).

For example:

**SpiderRedirectReplaceCSVs=/value1/,/value2/,/more\_values/**

**SpiderRedirectReplaceWithCSVs=/new\_value1/,/new\_value2/,/more\_new\_values/**

For example:

**SpiderRedirectReplaceCSVs=/value1/,/value2/,/more\_values/**

**SpiderRedirectReplaceWithCSVs=/new\_value1/,/new\_value2/,/more\_new\_values/**

In this example, the URL's **/value1/** string is replaced with the **/new\_value1/** string, the **/value2/** string is replaced with the **/new\_value2/** string, and the **/more\_values/** string is replaced with the **/more\_new\_values/** string.

## [Spider] section

This section summarizes the spiders that you have defined in the [**<MySpider>**] sections.

### ImportPath

Specify the path to the location where IDX files are created. This is usually the same as the location specified for **IndexPath**.

### IndexPath

Specify the path to the location from where IDX files are indexed into the DRE. This is usually the same as the location specified for **ImportPath**.

### Number

Specify the total number of spiders that you have defined in the [**<MySpider>**] section.

### <n>=<MySpiderName>

#### <n>

Enter the number of the spider. Note that the numbering must start from **0** and be sequential.

#### <MySpiderName>

Enter the name of the spider. All spiders that you are listing must refer to an actual spider that you have defined in a [**<MySpider>**] section. For example:

0=<My\_first\_spider>

1=<My\_second\_spider>

## **[<MySpider>] section**

You should replace this section with the spiders you want the Fetch to run. Any settings that you specify for a spider will override the default settings (but only for this spider).

For example:

**[My\_first\_spider]**

**Settings for the first spider**

**[My\_second\_spider]**

**Settings for the second spider**

**Note:** by default, the configuration file contains an **[Autonomy]** spider.

## **Typical Configurations**

The typical installation of HTTPFetch consists of a single instance running on a single machine. In this case the parameters used for the fetch procedure are stored in a file residing in the same directory as the HTTPFetch application executable. The configuration file will have the same name as the executable, but the extension “.cfg” instead of “.exe”.

In the typical installation case, the installation script provides the option of configuring which web sites are to be indexed. If this option was selected then the configuration file will be populated with some sections describing how to fetch the specified sites. If not, some example sections will be present in the configuration file.



## Example HTTPFetch configuration file

The following is an example of an HTTPFetch configuration file:

The number of sections in your configuration file depends on the number of spiders that you have. In this example, the Fetch runs at 3am every day by default and there are two spiders. The spiders are called **AUTONOMY** and **MyCompany**. Each fetch can download 2000 pages per fetch cycle.

```
[LICENSE]
Holder=MyCompany
Key=4179999624992961901307N9820011

[SERVICE]
ServicePort=40010
ServiceControlClients=127.0.0.1
ServiceStatusClients=127.0.0.1

[DEFAULT]
DREHost=127.10.9.2
IndexPort=2001
Database=News
LogFile=spider.log
nSockets=16
SpiderRepeatSecs=86400
SpiderCycles=-1
Depth=99
SpiderStartTime=03:00
SiteDuration=43200
MaxPages=2000
ImportStripLinks=true
FollowRedirect=true
StayOnSite=true
MinPageSize=4090
MaxPageSize=163840
MaxLinksPerPage=100
PageTimeout=100
CantHaveCheck=129
CantHaveCSVS=*archive*
AfterDate=-365
BeforeDate=7
SpiderAs=HTTPFetch Spider, contact spider@MyCompany.com for details
DateFormats=DDMMYYYY,YYMMDD,YYMMD,DDMONTHYYYY,LONGMONTH
DD,MMDDYY,MMDD
BatchProcess=IMPORT
BatchSize=200
```

## Configuring HTTPFetch

```
ImportMetaToFields=true  
FollowRobotProtocol=true  
PageDelay=30
```

```
ImportSummary=true  
ImportStoreContent=true  
IndexMode=REFERENCE  
IndexOverSocket=true  
StoreSiteStructure=true
```

```
ImportBreaking=true  
ImportBreakingMinParagraphWords=300  
ImportBreakingMaxParagraphWords=500  
ImportBreakingMinDocWords=500  
ImportMinLengthWords=10  
ImportMinLength=100
```

```
[ SPIDER ]  
ImportPath=C:\Autonomy\HTTPFetch  
IndexPath=C:\Autonomy\HTTPFetch
```

```
Number=1  
0=AUTONOMY
```

```
[ AUTONOMY ]  
URL=http://www.autonomy.com/  
Directory=AUTONOMY  
LogFile=AUTONOMY.log
```

```
[ MyCompany ]  
URL=http://www.mycompany.com/  
Directory=MyCompany  
LogFile=MyCompany.log
```

## 5. HTTPFetch Administration Utility

---

Although you may still manually configure HTTPFetch, the Administration Utility provides a simple GUI for performing basic administration tasks. Additionally, by using it, you can test a particular spider's configuration before you activate it. This will catch many configuration problems before they happen.

Please note that if you wish to keep comments in configuration files used by the fetch GUI, enter them as comment0=whatever comment1=something else

Restart from zero in each section to avoid getting lost.

This is necessary because when the method used to read in configuration files reads them in, it only reads in what is specified in [section\_name] or key=value and it only writes the same out.

Throughout this chapter, you may find it helpful to refer to the explanations of the configuration parameters as they appear in the table in the configuration chapter.

### Starting the Administration Utility

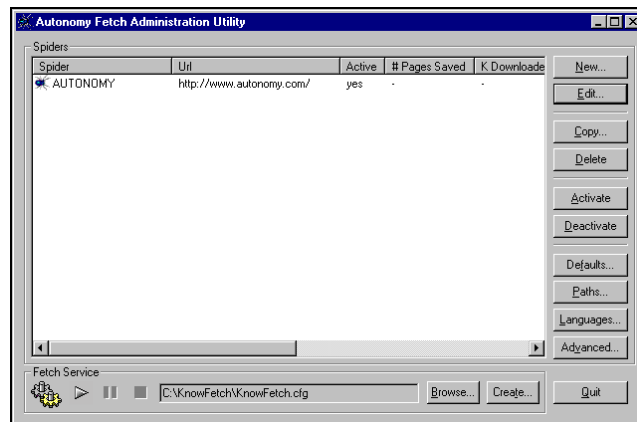
To start the HTTPFetch Administration Utility, click on your Windows **Start** Button. Then select Programs/<Installation Name>. In here you will find two applications: Configure the <Installation Name> and Uninstall the <Installation Name>. Click on the application called Configure the <Installation Name>. The HTTPFetch Administration Utility will then be launched displaying the various spiders that you have configured in your HTTPFetch Configuration File.

### Selecting an alternative instance of the HTTPFetch to configure

Should you wish to configure a HTTPFetch other than the one with which your Administration Utility was installed, it is possible to point your Administration Utility to this Alternative Configuration File. To do this, click on the **Browse** button and navigate to the required HTTPFetch configuration file. You can also use the HTTPFetch Administration Utility to generate configuration files for UNIX. Simply copy the configuration file to UNIX and change the paths and IP addresses it refers to once it is on the UNIX machine.

## The Main Window

The main Fetch Administration Utility window is divided into columns - Spider, URL, Active, Pages Saved and K Downloaded. The Spider column shows the unique name of each spider. The URL column displays the starting point for each spider. The Active column indicates whether or not the spider is active, i.e. the spiders that will be used the next time the fetch is run. This column contains a value of "Yes" if the spider is currently active, and "No" if it is currently disabled. The Pages saved column displays the number of pages saved and the K Downloaded shows how much data has been downloaded in Kilobytes.



The right-hand side of the window contains a series of buttons for performing various editing actions:

### New

This displays the New Spider Wizard, which guides you through the process of creating a new spider (described below). You can also create a new spider by double-clicking an empty section of the list.

### Edit

This displays Edit Spider property pages (described below) so that the selected spider's parameters can be edited. You can also edit a spider by double-clicking its entry in the list.

### Copy

This creates a copy of the currently selected spider. Clicking it brings up the Clone Spider dialog where you can enter the name of the clone spider.

You can then edit this spider as if you had created it using the New Spider Wizard. Note that all spiders must have unique names.

### Delete

This deletes the selected spider(s). Note: if you delete any spiders, you will not be able to recover the spider configuration information at a later stage unless you have backed up the configuration file.

### Activate

This adds the selected spider(s) to the list of active spiders.

The parameters **number=**, and **n=** in the configuration file will automatically be updated to include the spider.

### Deactivate

This removes the selected spider(s) from the list of active spiders.

In the Configuration file the parameters **number=**, and **n=** need to be updated to not include the spider.

### Defaults

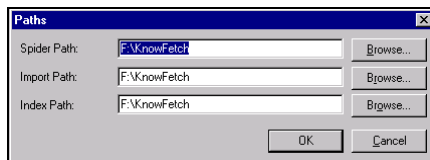
Clicking here will display the defaults. It displays the same set of property pages that are used for editing spider configurations, but allows you to edit the default values for any spiders subsequently created. This is useful if you wish to change any values across all subsequently created spiders. Note; it will not affect the values of any spiders already created.

In the Configuration file, the default parameters are listed under the **[Default]** section.

### Paths

This allows you to specify the directory paths where you want to store the files for importing and indexing. Clicking it displays the Paths dialog where you can enter the paths you wish to use.

You can browse the file system for each of the paths by clicking the **Browse** button.



In the Configuration file these parameters are **Spiderpath=** which is no longer used, **Importpath=** and **Indexpath=**. The parameters are found in the **[Spider]** section.

## Languages

This displays the **Edit Languages** dialog.

### New.

This displays the **Format Languages** dialog, which allows you to create a new language by specifying the language name, month names and abbreviated month names.

### Edit...

Click on Edit in the **Edit Languages** dialog to display the Format Language dialog, which contains the language name, month names and abbreviated month names of the selected language.

### Remove

This removes the selected language. Note: if you remove the language, all of the spiders configured to use it will still use that language, however if you re-edit them, they will default to using English. Once a language has been removed, you will have to recreate it by using the **New** button and entering the details again.

## Advanced

This displays the Advanced Parameters dialog, which allows you to specify the following parameters:

- The number of sockets you wish to use. Increasing this value increases the number of simultaneous connections that can be made and therefore speeds up the time it takes to run the fetch, however, it will also increase bandwidth and processor use, therefore a suitable value must be set depending on the bandwidth of the Internet connection and the speed of the machine that the fetch is ultimately to be run on. In the configuration file, **nSockets=**.
- The maximum rate at which to transfer data for all spiders. In the configuration file, **MaxKBPS=**.
- The interval between the start of successive fetches. In the configuration file, **SpiderRepeatInterval=** or **SpiderRepeatSecs=**.

<b>SpiderRepeatInterval will override SpiderRepeatSecs if both are specified</b>
--

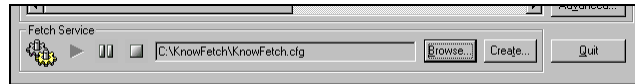
- The number of times you wish the spider to repeat. The value -1 can be entered here if you wish the spider to repeat indefinitely which can be useful if a site whose content changes every day is to be spidered. In the configuration file, **SpiderCycles=**.
- The time you wish to start the fetch. This can be changed by positioning the cursor in the hours or minutes field and increasing or decreasing the value with the up/down arrows situated next to the time display. In the configuration file, **SpiderStartTime=**.

### Fetch Configuration File

At the bottom of the window you will find a file system path (as shown below) which points to the configuration file associated with the current set of spiders displayed in the window.

If you installed the Administration Utility at the same time as the HTTPFetch, then it will display the path of the HTTPFetch configuration file. If you have multiple installations of the Fetch, you may configure each of them from the same Fetch Administration Utility by using Browse to locate the configuration file you wish to edit.

You can also create a new (empty) configuration file using **New** button. Both of these operations display the standard windows file selector dialog, which allows you to specify the name and location of the file you are creating/browsing for.



### Fetch Service

The three buttons in the bottom left hand corner of the main window are play, pause and stop respectively and you control the MAIN fetch service with these buttons - STOP stops the service, PLAY starts the service and PAUSE pauses the service.

### Browse

This allows you to browse for an existing configuration file. A standard Windows File Selector is displayed, allowing you to browse to the directory and select the existing filename.

### Create

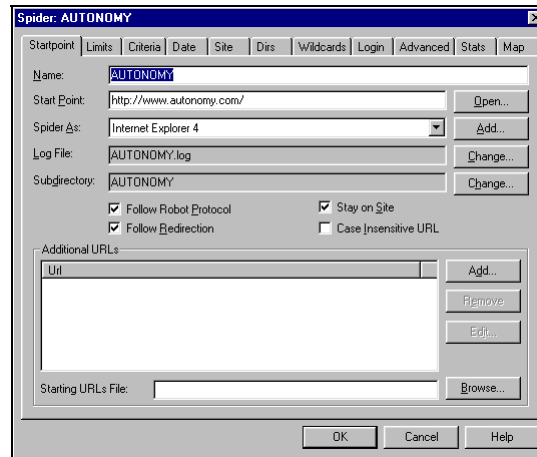
This allows you to create a new configuration file. A standard Windows File Selector is displayed, allowing you to browse to the directory and enter the new filename.

### Quit

The **Quit** button located on the bottom right-hand corner of the window will exit the application.

## The Startpoint Page

Double-clicking on the name of a spider in the main window will launch the startpoint page. The Startpoint Page controls all aspects of what should be spidered and where the content and logging data should be stored.



### Name

Name the spider by typing a descriptive name into the Name field. This field can include any alphanumeric character but no punctuation characters except spaces are allowed.

In the configuration file, **[Name]**.

### Start Point

This is the first URL the spider looks at either for content, or for additional links to follow, for example, you might wish to spider daily news from Yahoo! To do so, enter `http://dailynews.yahoo.com/headlines/` into the Start Point field. The URL you supply must be valid and complete (i.e. including the `http://` prefix). You can examine the URL you have entered by clicking the **Browse** button, which will open up the page in your default browser. It is a good idea to check that the URL you have specified points to the place that you think it does.

In the configuration file, **URL=**.



### Spider As

When a browser connects to a web site, it sends a user agent string, which identifies what type of browser it is. By default, the Fetch Administration Utility configures spiders to send a user agent string that identifies the spider as Microsoft's Internet Explorer 4.0. If you do not wish to spider as an IE 4 browser, you may select from pre-configured user agent strings, or you may specify your own by clicking the **Add** button. This allows you to enter the name of the user agent you wish to spider as. Internet Explorer 4 (Default). In the configuration file, **SpiderAs=**.

### Log File

Each spider outputs status information and errors to a log file. You may change the name of this file by clicking on the **Change** button, which allows you to enter the new file name. The default is the same as the spider name. In configuration file, **LogFile=**.

### Subdirectory

Each spider may create a working directory for its temporary files (depending on the spider mode you are using). The Subdirectory setting allows you to configure the location of this directory on a per-spider basis. You can change the setting to a directory of your choice if you wish by clicking the **Change** button, which allows you to enter a new directory name. The default is the same as spider name. In the configuration file, **Directory=**.

### Follow Robots Protocol

This checkbox tells the spider whether or not it should follow the Robots Protocol by looking for and following the directives in a file called robots.txt at each site it spiders (see <http://www.cnn.com/robots.txt> for an example). The Robots Protocol is a formalized way of excluding robots from a particular site, or parts of a site. Well-behaved 'bots will obey the directives found in this file. If you decide not to follow the robots protocol, you should take care to increase the Page Delay setting on the Limits Page. This is because you are downloading parts of a site that the owner does not want you to download and you will be using up a large number of the available connections to the site. This will slow down access for other users. It defaults as checked. In the configuration file, **FollowRobotProtocol=**.

### Follow Redirection

Redirection is when a page has been temporarily moved. When this happens, the HTTP server sends a 302: Temporarily Moved header, along with the new location of the page. This process is normally transparent to the browser however, you must decide whether your spider will honor this redirection or not. It defaults as checked. In the configuration file, **FollowRedirect=**.

### Stay on Site

This checkbox tells this spider whether or not it should follow links off site, or only follow links to pages that are located on this server (i.e. in this subdomain). Note: Although it is not intuitive, if you are pointing a spider at a given start point, and all of the links from that page point to other subdomains, you will not get any results back. For example, if your start point is `http://www.cnn.com/`, and you have chosen to stay on site, you will NOT be retrieving pages from `http://japan.cnn.com/`. Although they are in the same top level domain, these two sites are considered different by the spider. Perhaps a more apt way to think of the Stay on Site setting is to think of it as Stay on Server. NavSiteAllowed settings can also be used to restrict to any parts of a domain. Stay on Site defaults as checked. In the configuration file, **StayOnSite=**.

### Case Insensitive URL

This option allows you to work transparently with NT-based WWW servers. Although the HTTP standard specifies that URLs are case sensitive, certain NT-based HTTP servers (specifically, all versions of Microsoft's Internet Information Server) fail to follow this standard. If you select this option, for example, the `http://www.company.com/dir/doc.html` and `http://www.company.com/DIR/DOC.html` will be treated as if they are the same. If you know that the site you are spidering is case insensitive, this option will save you from having duplicate content indexed into the DRE. Note that on all flavors of UNIX, the above examples will be unique documents. It defaults as unchecked.

### Additional URLs

In addition to the Start Point URL, you may specify a list of additional start Point URLs to be spidered. As with the actual start point, these URLs must be valid, complete (non-relative) URLs. You may add an additional URL by selecting the **Add** button or double-clicking an empty space in the list. To remove a URL, select it and click the **Remove** button. To edit a URL, either select it and click **Edit** or double-click on the URL from the list. Note that if you specify URLs here, which contain a different domain name to the start point URL, you must not check the Stay On Site box, otherwise the sites will not get spidered.

In the configuration file, **URL0=**.

### Additional URLs File

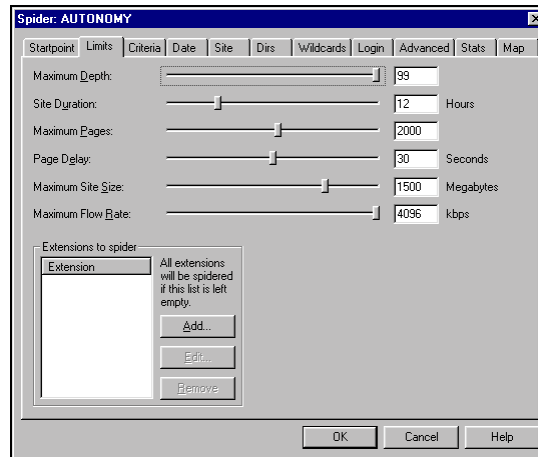
If you want to include a lot of URLs in the spider, you may not want to type them in one by one. In this case, you may specify a file containing URLs using the Additional URLs File field. Use a plain ASCII text file that contains each of the URLs you want the spider to examine, separated by a new line character. For example:

```
http://www.autonomy.com/  
http://support.autonomy.com/  
http://www.usatoday.com/news/washdc/nc1.htm
```

You can either enter the name of the file manually, or press the **Browse** button which will display the standard Windows File Selector to allow you to browse for the file you wish to use. In the configuration file, **URLFile=**.

## The Limits Page

The Limits Page allows you to specify the upper limits on the amount of information your spider should retrieve from a given site (or set of sites). This page is shown below:



### Maximum Depth

This parameter specifies how many levels of links the spider should follow. Like a file system, web sites follow an inverted tree structure. The top-level page (usually index.html or something similar) contains links to other pages, which in turn each contain links to more pages. The top-level page is level 0, each of the pages which the top-level page links to are level 1, each of the pages which level 1 links to are level 2, and so on. **Default: 3.**

Needless to say, setting a large maximum depth may result in an extremely large number of pages being retrieved, so set the maximum depth to something small enough to ensure that you don't download the entire Internet, but large enough to ensure that you get the content you are really interested in. In the configuration file, **Depth=**.

### Site Duration

This parameter tells the spider how long, in hours, it should spend on a given site (here defined as the starting URL, plus all of the valid links leading away from it, up to the depth specified in maximum depth, above). **Default: 1 hour.** In the configuration file, **SiteDuration=**.

### Maximum Pages

This parameter specifies how many documents the spider should retrieve from the spidered site. **Default: 2000.** In the configuration file, **MaxPages=**.

### Page Delay

This setting specifies how long, in seconds, that the spider should wait on average after it has downloaded a page before it downloads the next link or set of links.

It should be noted that the page delay is an average value which covers time taken for downloading the page and waiting time. This means that if page delay is set to 60, then in 600 seconds about 10 pages will be downloaded although some will have been downloaded more quickly than others. If the spider is operating too quickly, then it will slow down so that the average value is 60.

It is not necessary for the first page to be fully downloaded before the links for the next depth to be commenced. Although the default page delay is 0, you should take care to increase this to a reasonable number given the bandwidth and number of concurrent users which the site you are spidering can support. If you do not give other (human) users a chance to connect to the remote site, you will make yourself extremely unpopular with that site's administrators. **Default: 0.** In the configuration file, **PageDelay=**.

### Maximum Site Size

This parameter is used to tell the spider how much content it should retrieve. It is useful to avoid filling your hard disk - it can sometimes be difficult to determine in advance just how much content may come back to you if the maximum values set above are followed. You may want to reduce the default value based on how much disk space you have available. **Default: 1.5Gb.** In the configuration file, **TotalSize=**.

The totalsize setting can be in Gigabytes, Megabytes or Kilobytes:

TotalSize=10gbytes

TotalSize=10000mbytes

TotalSize=10000000kbytes

The total maximum is 2 TBytes (Terabytes) which can only be set by using gbytes/mbytes/kbytes and not through the default bytes setting.

### Let me specify extensions to spider

This parameter controls the file extensions HTTPFetch will download. The default behavior is to download everything, but you may wish to download only html-type documents, in which case you would use the extensions box as described below. **Default: unchecked.**

**Extensions**

You should make a list of the extensions which you wish to have spidered in the box provided, for example:

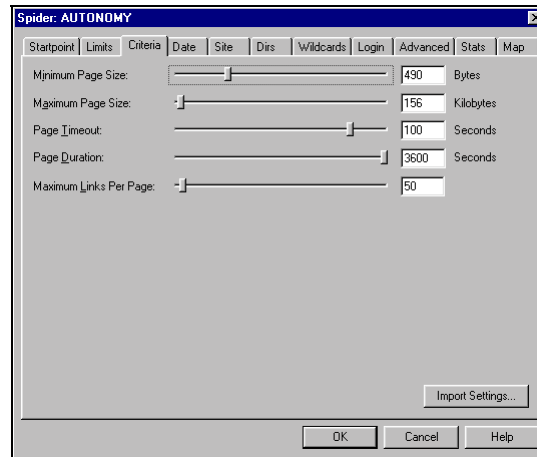
- \*.htm
- \*.html
- \*.shtml
- \*.fhtml
- \*.asp
- \*.cfm

This will retrieve all files with the above extensions. You can specify the extensions in the configuration file using a comma separated list. The parameter for this is **Extensions=**.

NB To use more than one wildcard, for example\*.htm\* which would take '.htm', '.shtml' etc, you must enter it in the wildcards page of the Fetch Admin not in the list of extensions

## The Page Criteria Page

The Page Criteria page allows you to set limits on the amount of information your spider should retrieve from a given site (or set of sites) and the amount of time it should wait/take in performing certain operations. This page is shown below:



### Minimum Page Size

The minimum page size in bytes that is to be downloaded. If the page is smaller than this it is ignored. **Default: 3 bytes.** In configuration file, **MinPageSize=**.

### Maximum Page Size

The maximum page size in bytes that is to be downloaded. If the page is larger than this then it is ignored. **Default: 1000000 bytes (approx. 1 Mb).** In configuration file, **MaxPageSize=**.

### Page Timeout

The period of time that the spider will wait after requesting a page from the server before receiving any data. If the spider has to wait longer than this, then the attempt to fetch that particular page is cancelled. This parameter is not to be confused with Page Duration (see below). **Default: 100 seconds.** In configuration file, **PageTimeout=**.

**Page Duration**

The maximum time that the spider should spend downloading any individual page. If the time exceeds this value, then the page download is terminated. Be careful not to set this parameter too low since any connections over low bandwidth/high traffic links may take a long time. **Default: 3600 seconds (1 hour).** In configuration file, **PageDuration=**.

**Maximum Links Per Page**

The maximum number of links a page may contain before not being downloaded. It is useful, since pages with a high number of links on them are more likely to be index pages and may well not contain any valuable content. **Default: 50.** In configuration file, **MaxLinksPerPage=**.

**Import Settings**

Launches the import wizard (see next page).

## Import Parameter Settings

To access the import parameters for each spider, click on the **Import Settings** button in the **Criteria** page to launch the Import Wizard. The Import wizard allows you to edit the import keys.

Note:

you can also run the Import Wizard manually. There is an import wizard (executable) which edits the import keys of a configuration file. When you run it, it displays the arguments required in a popup message box. These are as follows:

(-f<filename> -s<section> -c<caption> -p<pagenumber> -n -v)

<filename>	the name of the file which you wish to edit
<section>	the section of the file (can be the default section if desired)
<pagenumber>	the page number to open the wizard at (0 based index)
-n	runs a new wizard, instead of the edit pages(default)
-v	displays the version number in a popup message box and then exits.

NB:

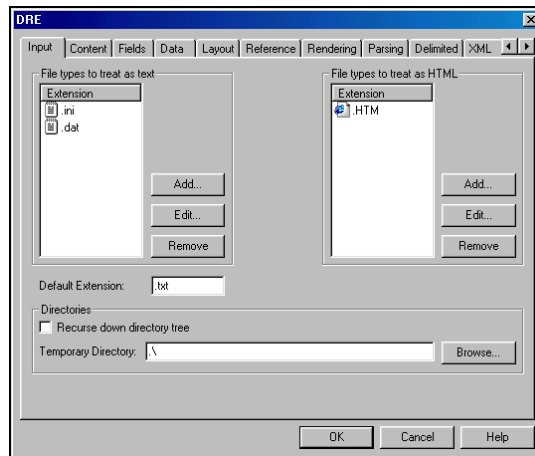
Any parameters with spaces must be delimited by 's

Do not put spaces between the flag and the value (i.e. -fc:\directory\fetch.cfg not -f c:\directory\fetch)

Anyone needing an import wizard for their program should use this one. You can run it in C, C++, VB by calling the windows API function ShellExecute.



## Input



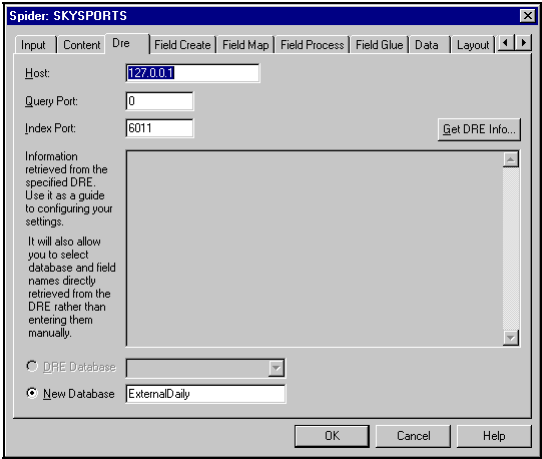
Parameter	Description
File types to treat as text (TextImportExtns=)	File types that will be imported as though they are text files. You can add and remove extensions from the list using the corresponding buttons.
File types to treat as HTML (HTMLImportExtns=)	File types that will be imported as though they are HTML files. You can add and remove extensions from the list using the corresponding buttons.
Default Extension (ImportDefaultExtensions=)	Specifies the extension by which all unrecognized file types are to be masked as.
Directories (IMPORTTEMPDIR=)	Specifies whether or not to import documents that appear in a subdirectory of the working directory.

## Content

Parameter	Description
Store Plain Text Content ( <b>ImportStoreContent=</b> )	Specifies whether or not the text content of the documents are to be imported
Store Summary ( <b>ImportSummary=</b> <b>ImportSummarySize=</b> )	Specifies whether or not the first n lines of the documents are to be imported as a summary.
Intelligent Title and Summary ( <b>IMPORTINTELLIGENTTITL</b> <b>ESUMMARY=</b> )	Specifies whether or not a unique intelligent title and summary are to be generated for the documents that are imported.
Minimum Characters in Title ( <b>ImportMinTitleChars</b> )	This specifies the minimum amount of characters that can be used as a title. If a title of less than 3 characters is found the title will be taken from the content of the document.
Criteria – Min/Max Words ( <b>ImportMinLengthWords=</b> <b>ImportMaxLengthWords=</b> )	Indicates that documents with less than Min words and more than Max words are not to be imported.

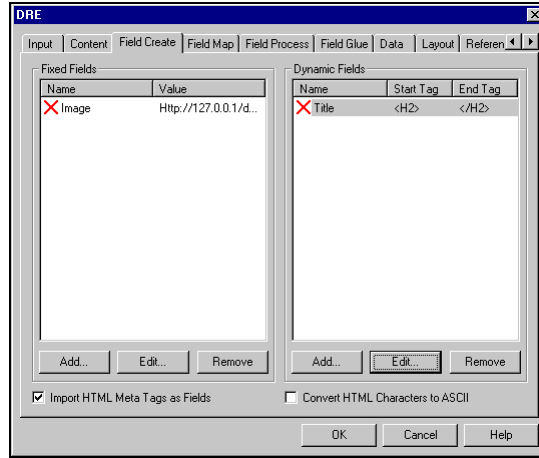
Parameter	Description
Criteria – Min/Max Bytes <b>(ImportMinLength= ImportMaxLength=)</b>	Indicates that documents with less than Min Bytes and more than Max Bytes are not to be imported.
Breaking – Break large documents <b>(ImportBreaking=)</b>	Specifies whether or not documents over a certain size should be broken up into smaller sections for easier indexing.
Breaking – Break if over <b>(ImportBreakingMinDocWords=)</b>	If documents are over this number of words, then the document is broken up into smaller sections for easier indexing.
Breaking – Break into paragraphs between <b>(ImportBreakingMinParagraph Words= ImportBreakingMaxParagraph Words=)</b>	When documents are broken up into smaller sections, the sections have at least n paragraphs and at most m paragraphs.

DRE



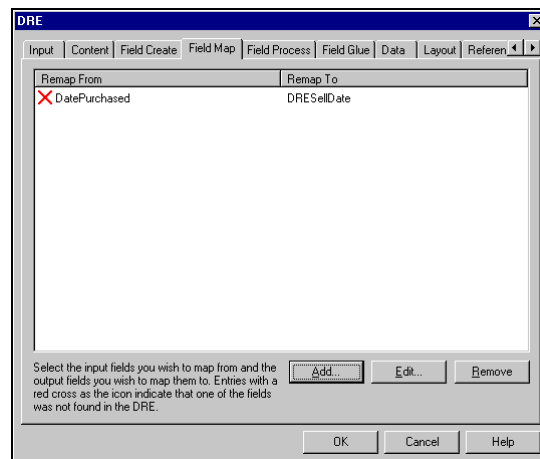
Parameter	Description
Host (DREHost=)	IP Address of the machine which hosts the indexing DRE.
Index port (IndexPort=)	Index port of the indexing DRE.
Database (Database=)	Database into which documents obtained by the spider will be indexed.

## Field Create



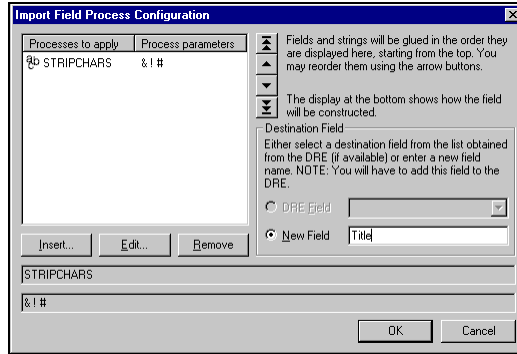
Parameter	Description
Fixed Fields – Name (ImportFixedFieldName=)	Name of the field in the DRE that the value in the document is to be stored in.
Fixed Fields – Value (ImportFixedFieldValue=)	Value that is to be stored in the DRE field. This value will be the same for all documents imported in this session.
Dynamic Fields – Name (ImportFieldName=)	Name of the field in the DRE that the value in the document is to be stored in.
Dynamic Fields – Start Tag (ImportFieldStartn=)	Set of characters that are to be used to denote the start of the value that is to be stored in the DRE field. This value can potentially be different for each document imported in this session.
Dynamic Fields – End Tag (ImportFieldStopn=)	Set of characters that are to be used to denote the end of the value that is to be stored in the DRE field. This value can potentially be different for each document imported in this session.
Import HTML Meta Tags as Fields (IMPORTMETATOFIELDS=)	Specifies whether or not the values of HTML meta tags are to be stored in DRE fields of the equivalent name. These fields must be specified in the DRE.INI before the documents are indexed.
Convert HTML Characters to ASCII (ImportFieldHTMLConvertChars=)	Specifies whether or not to convert HTML characters to ASCII. For example, &amp; would be converted to &.

## Field Map



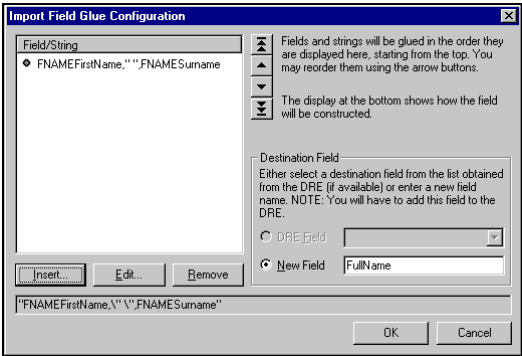
Parameter	Description
Field Remapping – Remap From (ImportRemapFieldFrom=)	Specifies the name of a field whose value is to be taken and inserted as a value for an alternative field.
Field Remapping – Remap To (ImportRemapFieldTo=)	Specifies the name of a field whose value is to be the same as a value in an alternative field.

## Field Process



Parameter	Description
Process to Apply (ImportFieldOpn)	This specifies the name of the operation to apply to the specified field See import parameters for list of available entries here.
Process Parameters (ImportFieldOpParamn)	This specifies the parameters to go with the operation/process being applied to the specified field See import parameters for list of available entries here.
New Field (ImportFieldOpApplyTon)	This is the Field to apply the operations to.

## Field Glue



Parameter	Description
Destination Field (New Field) (ImportFieldGlueDestination)	This is the destination field to glue the source csvs into. See the Import Parameters for more details.
Source CSVs (ImportFieldGlueSourceCSVs)	This allows fields and or strings to be glued together. Specify FNAME<FieldName> and / or a string. See Import Parameters for more details.



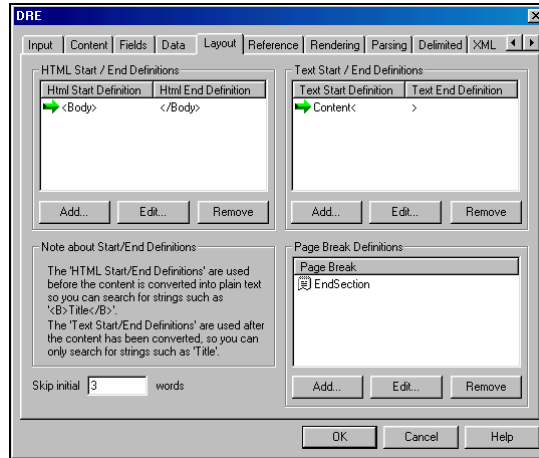
## Data

The screenshot shows the 'DRE' dialog box with the 'Data' tab selected. It contains two main sections: 'Date' and 'Length'. In the 'Date' section, 'Extract from:' is a dropdown menu set to 'Field', 'Extract from field:' is a text box containing 'NewDate', 'Extract to field:' is a text box containing 'DREDATE', and 'Extract to format:' is a text box containing 'dd/mm/yyyy'. To the right of these is a large empty 'Format' text area. Below the 'Date' section is the 'Length' section, where 'Extract from:' is a dropdown menu set to 'File' and 'Extract to field:' is a text box containing 'FILELENGTH'. Between the 'Date' and 'Length' sections are three buttons: 'Add...', 'Edit', and 'Remove'. At the bottom of the dialog are 'OK', 'Cancel', and 'Help' buttons.

Parameter	Description
Date – Extract From ( <b>IMPORTEXTRACTDATEFROM=</b> )	Indicates from where in the File data a date is to be extracted. This can be either: <ul style="list-style-type: none"> <li>- Now          Current date</li> <li>- Accessed    Date it was last accessed</li> <li>- Created      Date it was created</li> <li>- Modified    Date it was last modified</li> <li>- Field        Date taken from a specific file field</li> <li>- Content     First date found in the content of the document</li> <li>- Filename    First date found in the file name</li> </ul>
Date – Format ( <b>IMPORTEXTRACTDATEFORMA TCSVS=</b> )	Specifies the formats in which the required date can be found.
Date – Extract from field ( <b>IMPORTEXTRACTDATEFROMFI ELD=</b> )	Specifies the name of a date field in the DRE whose value is to be extracted.

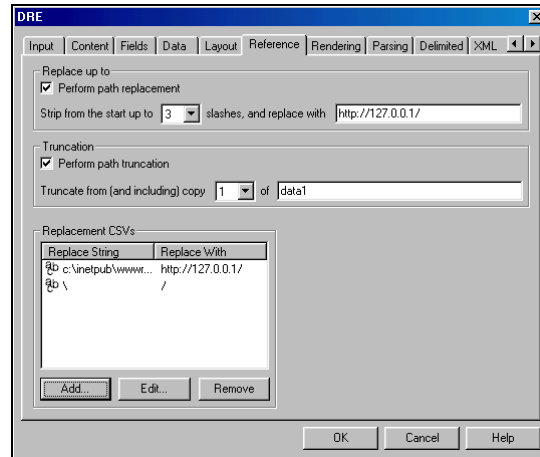
Parameter	Description
Date – Extract to field ( <b>IMPORTEXTRACTDATETOFIELD=</b> )	Specifies the name of the DRE field where the date extracted is to be stored.
Date – Extract to format ( <b>ImportExtractDateToFormat=</b> )	Specifies the format that the extracted date will be stored as in the DRE. If you are extracting to the field DREDATE, then the format must be yyyy/mm/dd or number of seconds since 1970 or NOW
Length – Extract from ( <b>IMPORTEXTRACTLENGTH=</b> )	Specifies from where the length of the document is to be extracted. Currently this can only be "File".
Length – Extract to field ( <b>ImportExtractLengthToField=</b> )	Specifies the name of the field in the DRE where the length of the document is to be stored.

## Layout



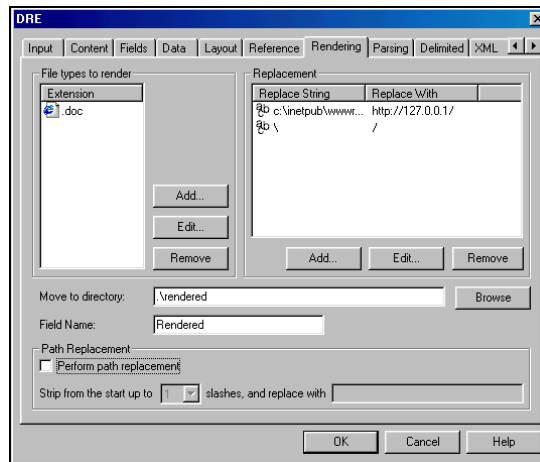
Parameter	Description
HTML Start/End Definitions – HTML Start Definition (HTMLIMPORTSTARTDEFCSVS=)	A string of characters that mark the start of the content to be imported in HTML documents.
HTML Start/End Definitions – HTML End Definition (HTMLIMPORTENDDEFCSVS=)	A string of characters that mark the end of the content to be imported in HTML documents.
Text Start/End Definitions – Text Start Definition (IMPORTSTARTDEFCSVS=)	A string of characters that mark the start of the content to be imported in text documents.
Text Start/End Definitions – Text End Definition (IMPORTENDDEFCSVS=)	A string of characters that mark the end of the content to be imported in text documents.
Page Break Definitions (IMPORTPAGEBREAKDEFS=)	Specifies a string that is used to mark a document break. This is used for splitting up documents into segments with each segment being contained in one idx format. When the idx files have been indexed into the DRE, the individual segments are joined back together to produce the original document.
Skip initial <i>n</i> words (ImportStartSkipWords=)	Specifies that the first <i>n</i> words in a document are to be skipped when importing the document content.

## Reference



Parameter	Description
Replace up to – Perform Path Replacement	Specifies whether or not the string up to a certain \ in the file reference is to be replaced.
Replace up to – Strip from the start up to... (ImportPathReplaceUpToSlash=)	Specifies the nth slash where the preceding string is replaced with the string specified.
Replace up to – ...and replace with (ImportPathReplaceString=)	Specifies a string that is to replace the string preceding the nth slash in a document reference.
Truncation – Perform path truncation	Specifies whether or not to truncate the file reference for imported documents.
Truncation – truncate from copy (ImportRefTruncateAfter=)	Specifies that the reference must be truncated after the nth occurrence of the string specified.
Truncation – ...of... (ImportRefTruncateString=)	Specifies the string that is used to truncate the reference after the nth occurrence.
Replacement CSVs – Replace string (ImportRefReplaceCSVs=)	Specifies the string that is to be replaced.
Replacement CSVs – Replace with (ImportRefReplaceWithCSVs=)	Specifies the string that is to replace the original reference of the file.

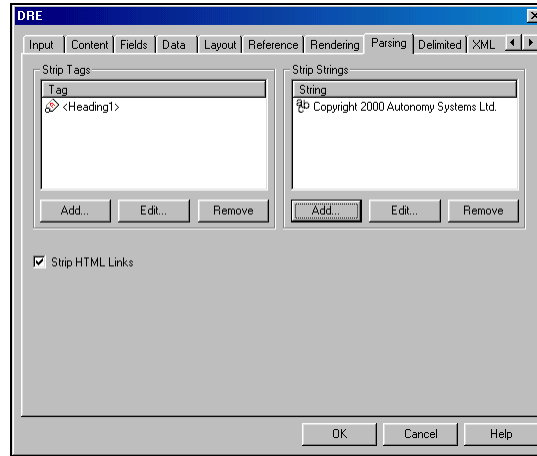
## Rendering



Parameter	Description
File types to render – Extension ( <b>ImportRenderedHTMLExtensions=</b> )	Specifies the extension of files on which to perform HTML rendering
Replacement – Replace string ( <b>ImportRenderedHTMLRefReplaceCSVs=</b> )	Specifies the string that is to be replaced. Used to set the right reference for the Field Name field.
Replacement – Replace with ( <b>ImportRenderedHTMLRefReplaceWithCSVs=</b> )	Specifies the string that is to replace the original reference of the ImportRenderedHtmlFieldName field. Used to set the right reference for the Field Name field.
Move to directory ( <b>ImportRenderedHTMLMoveToDir=</b> )	For any non-HTML file that gets imported, the HTML file that comes out of omnislave/PDF gets moved into the specified directory.
Field Name ( <b>ImportRenderedHTMLFieldName=</b> )	For any non HTML file that gets imported, the HTML file that comes out of omnislave/PDF gets moved into the this DRE field.

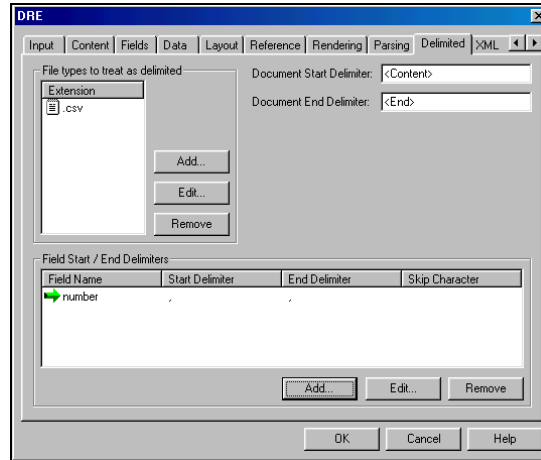
Parameter	Description
Path Replacement – Strip from the start up to ... <b>(ImportRenderedHTMLPathReplaceUpToSlash=)</b>	Specifies a string that is to be replaced up to a certain '\'. Used to set the right reference for the Field Name field.
Path Replacement – ...replace with... <b>(ImportRenderedHtmlPathReplaceString=)</b>	Specifies the replacement string of the string immediately preceding the nth '\'.

## Parsing



Parameter	Description
Strip Tags ( <b>ImportStripTagCSVs=</b> )	Strips the content between begin and end tags. E.g. <code>ImportStripTagCSVs=&lt;B&gt;</code> , will strip all content between <code>&lt;B&gt;</code> and <code>&lt;\B&gt;</code> .
Strip Strings ( <b>ImportStripStringCSVs=</b> )	Comma separated list of strings that, when matched in a document, are to be stripped from the idx format.
Strip HTML Links ( <b>IMPORTSTRIPLINKS=</b> )	Specifies whether or not to remove all content that is a hypertext link.

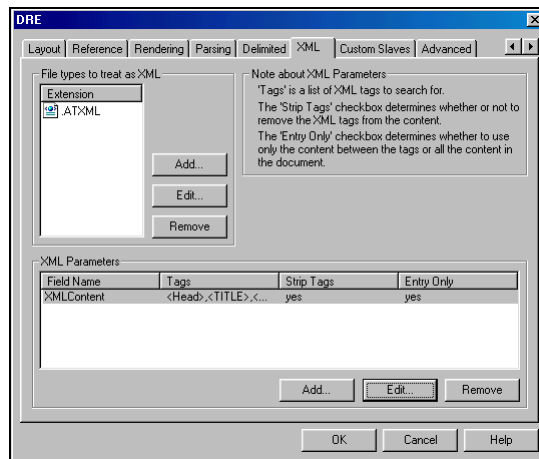
## Delimited



Parameter	Description
File types to treat as delimited ( <b>ImportDelimitedExtns=</b> )	Specifies the extension of the files to treat as delimited files.
Document Start Delimiter ( <b>ImportDelimitedDocStart=</b> )	In the case of importing a delimited file, e.g. a .csv file, this specifies the beginning of a file.
Document End Delimiter ( <b>ImportDelimitedDocEnd=</b> )	In the case of importing a delimited file, e.g. a .csv file, this specifies the end of a file.
Field Start/End Delimiters – Field Name ( <b>ImportDelimitedFieldn=</b> )	Specifies the name of the field where data from a delimited file is stored.
Field Start/End Delimiters – Start Delimiter ( <b>ImportDelimitedStartn=</b> )	Specifies the start string of data to be inserted into the idx field denoted by Field Name.
Field Start/End Delimiters – End Delimiter ( <b>ImportDelimitedEndn=</b> )	Specifies the end string of data to be inserted into the idx field denoted by Field Name.
Field Start/End Delimiters – Skip Character ( <b>ImportDelimitedSkipCharsn=</b> )	Specifies the number of characters to skip from the beginning of a document when producing the idx file.



## XML

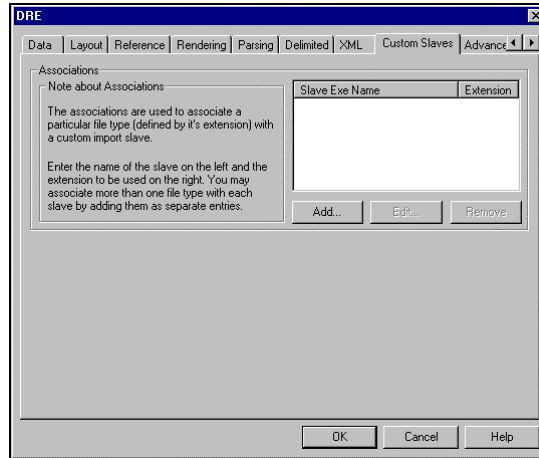


Parameter	Description
File types to treat as XML (ImportXMLExtns=)	A list of all XML file extensions.
XML Parameters – Field Name (ImportXMLFieldn=)	Will specify the field in the DRE where the content taken from the XML File is stored.
XML Parameters – Tags (ImportXMLSearchCSVTagsn=)	Searches for the appropriate XML tag(s). This can be a hierarchy level of search tags. For example, ImportSMLSearchTags0=body,head,title will take the content between the tags:  <body></body>, <head></head> and <title></title>.
XML Parameters – Strip Tags (ImportXMLStripTagsn=)	If set to yes, will strip the strings between the specified XML tags upon import of the file.

Parameter	Description
XML Parameters – Entry Only ( <b>ImportXMLEntryOnly=</b> )	<p>Only imports the content between the searched tags at that level.</p> <p>For example if an XML file contained the following strings:</p> <pre>&lt;TAG1&gt;   &lt;TAG3&gt;     Some Other Tag3 entry   &lt;/TAG3&gt;   &lt;TAG2&gt;     Some data   &lt;TAG3&gt;     The entry for Tag3   &lt;/TAG3&gt; &lt;/TAG2&gt; &lt;/TAG1&gt;</pre> <p>and</p> <p>ImportXMLSearchCSVTags0=TAG1,TAG2</p> <p>ImportXMLEntryOnly0=TRUE, then the content imported = “Some data”</p> <p>If ImportXMLEntryOnly0=FALSE, then the content imported = “Some data The entry for Tag3”</p>

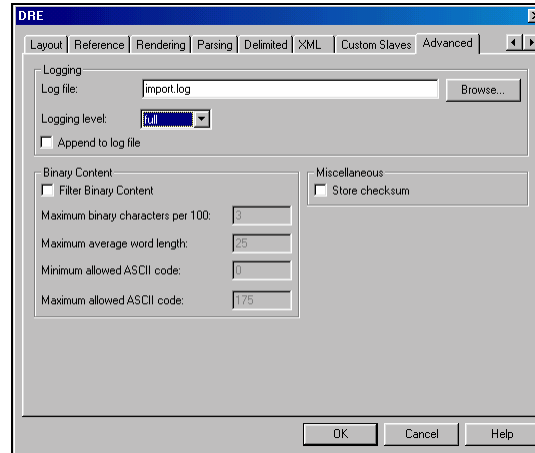
For more example of XML importing see the Import Parameters

## Custom Slaves



Parameter	Description
Slave Exe Name (ImportRegisterExecutable0=)	Name of the custom slave executable.
Extension (ImportRegisterExtnCSVs0=)	Extension of files that are to be processed by the custom slave executable.

## Advanced



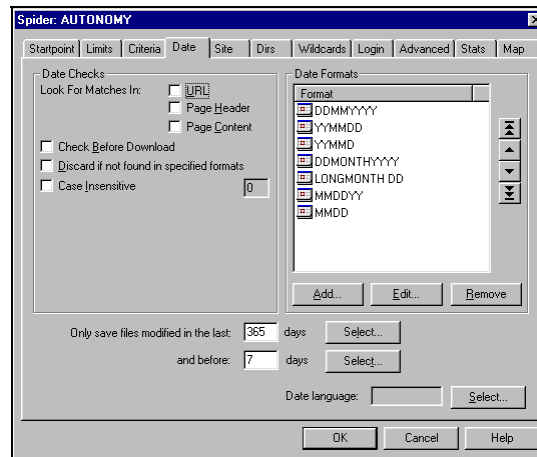
Parameter	Description
Log File ( <b>ImportLogFile=</b> )	Name to be given to the import log file
Logging Level ( <b>ImportLogLevel=</b> )	This is the type of logging that is to be performed. This can be either: <ul style="list-style-type: none"> <li>- Warnings</li> <li>- Errors</li> <li>- Full</li> </ul>
Append to log ( <b>ImportLogFileAppend=</b> )	Specifies whether or not import logging for this session should be appended to an existing log file.
Binary Content – Filter Binary Content ( <b>ImportFilterBinaryContent=</b> )	Specifies whether or not binary content is to be processed.
Binary Content – Maximum binary characters per 100 ( <b>ImportMaxBinaryCharsPerHundred=</b> )	Maximum number of binary characters tolerated per 100 characters. Anything binary exceeding this is to be ignored.

Parameter	Description
Binary Content – Maximum Average Word length ( <b>ImportMaxAverageWordLength=</b> )	Binary words with more than this number of characters will not be imported.
Binary Content – Minimum allowed ASCII code ( <b>ImportMinAllowedASCII CODE=</b> )	The minimum ASCII value allowed in characters in the binary content. Anything lower is treated as binary and removed.
Binary Content – Maximum allowed ASCII code ( <b>ImportMaxAllowedASCII CODE=</b> )	The maximum ASCII value allowed in characters in the binary content. Anything higher is treated as binary and removed.
Store Checksum ( <b>ImportChecksum=</b> )	When checked, a value is added to the Checksum field in the [Field] section in the DRE.INI. This field value is used to determine whether or not to show a document result in the front-end. This field is typically used with the Knowledge Update.

Once you have added all your import parameters you can then click on the **OK** button, which will take you back to the Main Import Property Sheet of the DRE GUI Admin. From here you can select the **OK** button and the idx file will be created. This file will be displayed in the idx list of the Import – Index Property Sheet of the DRE GUI Admin. Once you have added all idx files that you need to index into your DRE, click the **Index into DRE Now** button and the process will be executed with the listed idx files.

## The Date Page

The Date Page allows you to specify what various date values are to be obtained, into which field the value needs to be added and in what format. If you do not specify where to look for the date (Date Checks), it will automatically look at the last modified date.



### Look For Matches In

Where to look for dates to be matched. These can be none or more of those described in the following table. **Default: URL.** In the configuration file, **DATECHECK=**.

#### URL

In the URL.

#### Page Header

The section of the document delimited by the **HEAD** tags.

#### Page Content

The section of the document delimited by the **BODY** tags.

### Check before download

Check for the date before downloading the file. (If this is not checked, the file is downloaded and then examined).

### Discard if not found in specified formats

The page will not be downloaded if the date is not found in one of the formats specified.

### Case Insensitive

The date match will not be sensitive to the case of the URL or date format.

### Date Formats

This list displays all of the formats that date matches will be looked for in the places specified in the “Look for Matches In” section.

Date formats can be added by clicking the **Add** button or double-clicking an empty space in the list, removed by clicking the **Remove** button and edited by clicking the **Edit** button or double-clicking the entry in the list.

There is a set of buttons on the right-hand side of the list of date matches, which allow you to move a selected date format up or down the list. This is useful since date formats are checked in the order that they appear in the list, from the top down.

In the configuration file, **DATEFORMATS=**.

### Only Save Files modified in the last

This allows you to specify the date after which files will be saved. If the date matched is before this, the file will be ignored. This setting is particularly useful for spidering sites whose content changes regularly and quickly goes out of date, for example news sites.

Pressing the **Select** button displays a calendar from which a date can be selected.

Note: this parameter is relative and so specifies a number of days relative to the current date (a negative number represents a date in the past and positive number represents a date in the future). **Default: -1**. In the configuration file, **AFTERDATE=**.

### and before

This allows you to specify the date after which files will be saved. If the date matches is after this, the file will be ignored.

Pressing the **Select** button displays a calendar (shown below) from which a date can be selected.

Note: this parameter is relative and so specifies a number of days relative to the current date (a negative number represents a date in the past and positive number represents a date in the future). **Default: 1**. In configuration file, **BEFOREDATE=**.

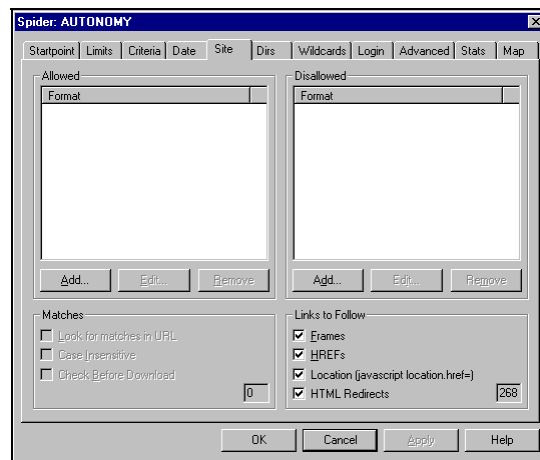
### Date Language

This allows you to select the language used for the date matches. Pressing the **Select** button displays the date selection dialog. This allows you to add, remove and edit date languages.

To select a language to be used, highlight the language and click “OK”. The name of the selected language will appear in the box next to the **Select** button. **Default: English**. In the configuration file, **DATELONGMONTHCSVS=**, **DATEMONTHCSVS=**.

## Site

This Window allows you to specify which sites must be visited and which sites must not be visited.

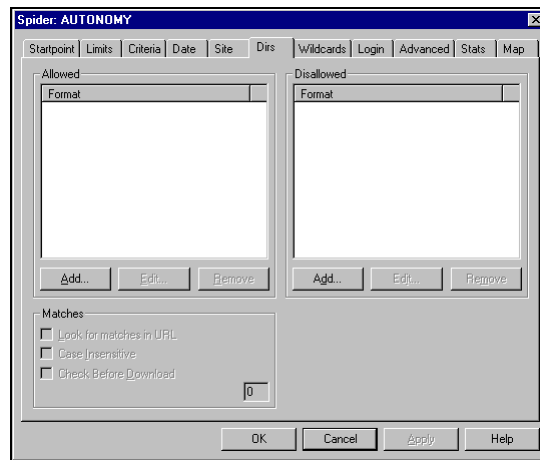


In the configuration file, **NAVSITECHECK=**, **NAVLINKSTOFOLLOW=**, **NAVSITEALLOWCSVS=**, **NAVSITEDISALLOWCSVS=**.



## Directory Specification

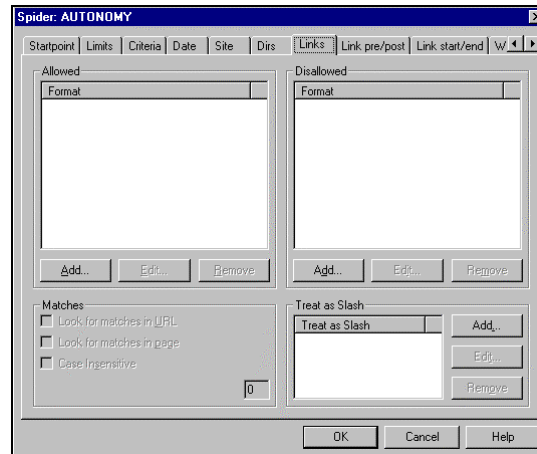
This window specifies which directories must and must not appear in the URL.



In the configuration file, **NAVDIRCHECK=**, **NAVDIRALLOWCSVS=**, **NAVDIRDISALLOWCSVS=**.

## The Links Page

This window allows you to enter values that the URL or content of a page must contain for the fetch to extract links from it.



### Allowed

This list displays strings that a page must contain for the fetch to extract links from it. You can add strings by clicking the **Add** button, remove strings by clicking the **Remove** button and edit strings by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **LinkAllowCSVs=**.

### Disallowed

This list displays strings that a page must not contain for the fetch to extract links from it. You can add strings by clicking the **Add** button, remove strings by clicking the **Remove** button and edit strings by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **LinkDisallowCSVs=**.

**Matches**

Allows you to specify where and how the fetch looks for the strings from the **Allowed** and **Disallowed** lists in pages. In the configuration file, **LinkCheck=**.

**Look for matches in URL**

The fetch looks in the URL of a page.

**Look for matches in page**

The fetch looks in the content of a page

**Case Insensitive**

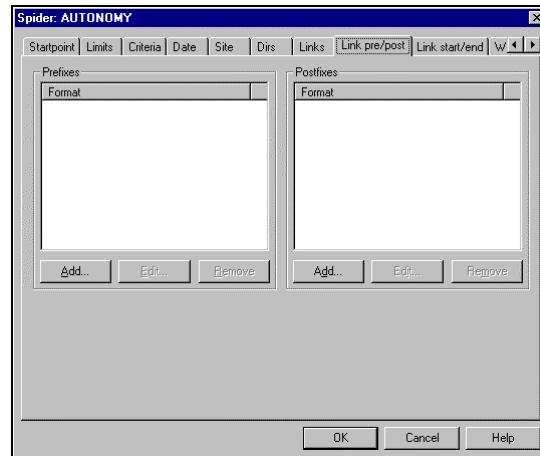
The fetch looks for case-insensitive matches.

**Treat as Slash**

Allows you to specify pages that you want to index as the root pages of sites. This list displays page names that the fetch strips from the URL of root pages of sites when indexing them into the DRE. You can add page names by clicking the **Add** button, remove page names by clicking the **Remove** button and edit page names by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **TreatAsSlash=**.

## The Link pre/post Page

This window allows you to specify prefixes and postfixes that the fetch adds to custom links when extracting them.



### Prefixes

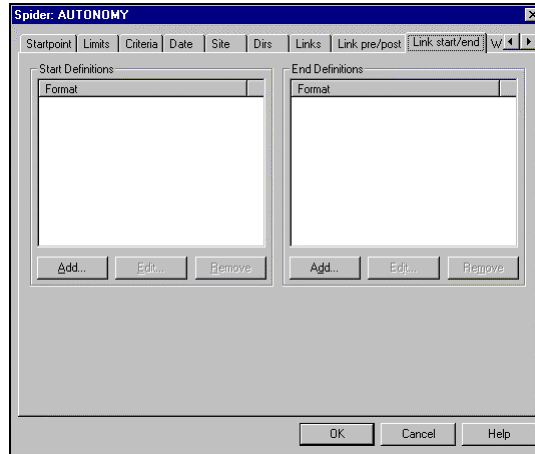
This list displays prefixes that the fetch adds to the start of custom links when extracting them. You can add prefixes by clicking the **Add** button, remove prefixes by clicking the **Remove** button and edit prefixes by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **NavLinkDefPrefixCSVs=**.

### Postfixes

This list displays postfixes that the fetch adds to the start of custom links when extracting them. You can add postfixes by clicking the **Add** button, remove postfixes by clicking the **Remove** button and edit postfixes by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **NavLinkDefPostfixCSVs=**.

## The Links start/end Page

This window allows you to specify strings that mark the start and end of custom links that are not written using the standard HTML format for links.



### Start Definitions

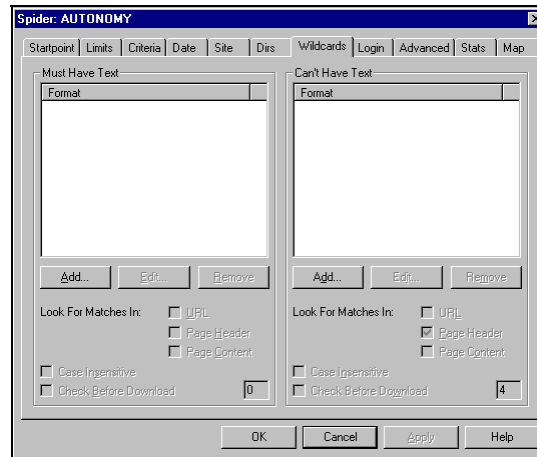
This list displays strings that mark the start of custom links in pages. You can add strings by clicking the **Add** button, remove strings by clicking the **Remove** button and edit strings by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **NavLinkDefStartCSVs=**.

### End Definitions

This list displays strings that mark the end of custom links in pages. You can add strings by clicking the **Add** button, remove strings by clicking the **Remove** button and edit strings by clicking the **Edit** button or by double-clicking the entry in the list. In the configuration file, **NavLinkDefStopCSVs=**.

## The Wildcards Page

The Wildcards Page allows you to specify text that must or must not be present in the page if it is to be used.



### Must Have Text

Expressions that must be contained in the section of the document that is being checked (as specified in the **Look for Matches** parameter). If a document does not contain the specified expression then it is not used (it may or may not be downloaded, depending on the status of the **Check Before Download** parameter). In the configuration file, **MUSTHAVECHECK=** and **MUSTHAVECSVS=**.

### Can't Have Text

Expressions that must not be contained in the section of the document that is being checked (as specified in the **Look for Matches** parameter). If a document contains the specified expression then it is not used (it may or may not be downloaded, depending on the status of the **Check Before Download** parameter). In the configuration file, **CANTHAVECHECK=** and **CANTHAVECSVS=**.

### Look for Matches In

See the description of this parameter in the **Date Page** section. **Default: URL**. In the configuration file, **MUSTHAVECHECK=** and **CANTHAVECHECK=**.

## The Login Page

The **Login Page** allows you to specify any additional login information that your spider may need to access a site.

Spider: AUTONOMY

Startpoint | Limits | Criteria | Date | Site | Dirs | Wildcards | **Login** | Advanced | Stats | Map

Method: **NOLOGIN** ☒ Encrypt username and password

URL:

Username Field:  Value:

Password Field:  Value:

Additional Login Information:

Name	Value

New... Edit... Remove

Cookies:

Name	Value

New... Edit... Remove

OK Cancel Help

### Method

This specifies the method that is necessary to login to the site. In general however, most sites don't require any additional login information, in which case the default **NOLOGIN** method is all that is required. A good method of determining whether it is necessary to enter login information to access a site is to browse to it and see if they require that you enter any additional personalization information, the first time you access the site.

#### **NOLOGIN**

No Login information needed.

#### **AUTHENTICATE**

Use authentication (name, password, etc.).

#### **FORM**

Fill in a form and send it back to the server.

#### **FORMGET**

Fill in a form and send it back to the server using the **GET** method.

#### **FORMPOST**

Fill in a form and send it back to the server using the **POST** method.

**Username Field**

The field name that the username is entered in.

**Username Value**

The username you wish to access the site with.

**Password Field**

The field name that the password is entered in.

**Password Value**

The password you wish to access the site with.

**Encrypt User Details Checkbox**

The username and password can be encrypted when written out to the configuration file by checking the "encrypt user details" checkbox."

**Additional Login Information**

This list allows you to enter any extra field name/value pairs that are needed to login to a site. Entries be added by clicking the **Add** button or double-clicking an empty space in the list, removed by clicking the **Remove** button and edited by clicking the **Edit** button or double-clicking the entry in the list. Doing so displays the following dialog, where you may enter the additional field/value pairs.

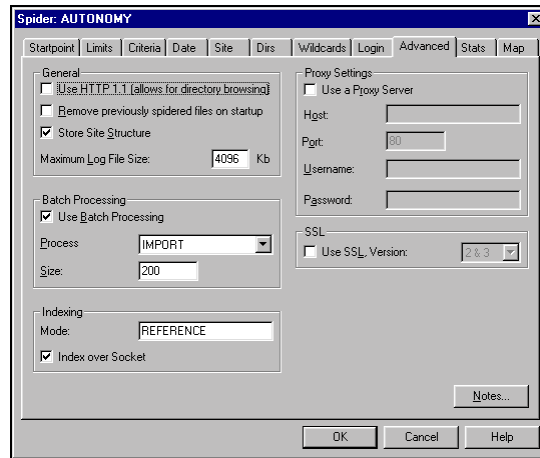
**Cookies**

This list allows you to enter any extra cookie name/value pairs that are needed to access a site. Entries can be added by clicking the **Add** button or double-clicking an empty space in the list, removed by clicking the **Remove** button and edited by clicking the **Edit** button or double-clicking the entry in the list. Doing so displays the **Spider Login Configuration** dialog where you can enter the additional field/value pairs.



## The Advanced Page

The **Advanced Page** allows you to specify the advanced spider parameters.



### General

#### Use HTTP 1.1

Specifies that you wish to use HTTP 1.1 instead of HTTP 1.0.

#### Remove Previously Spidered Files on Startup

Deletes the files that were created during a previous run of the spider.

#### Store Site Structure

Specifies whether or not to store the site structure

### Batch Processing

#### Use Batch Processing

Whether or not you wish to use batch processing.

#### Process

The batch process you wish to use – either **IMPORT** or **INDEX**.

#### Size

The size of the batch (number of documents) used during batch processing.

## Indexing

### Index Mode

Select the index mode which you want to use:

- |                         |   |
|-------------------------|---|
| <b>REFERENCE</b>        | Reference only.   |
| <b>MATCHNN</b>          | Conceptual match only (i.e. if the documents are similar above a certain threshold NN). |
| <b>REFERENCEMATCHNN</b> | Reference and then conceptual match as above.   |

### Index Over Socket

Specifies whether to index over the socket or not.

## Proxy Settings

### Use a Proxy Server

Check this box if you want to use a proxy server. The following fields are enabled:

- |          |   |
|----------|---|
| Host     | The address of the proxy server to use.                     |
| Port     | The port of the proxy server to use.                        |
| Username | The username to use to get access through the proxy server. |
| Password | The password to get access through the proxy server.        |

**Note:** If you want to access a web site that uses NTLM authentication, you can use the Advanced page to point the HTTPFetch at the NTLM Proxy module rather than at an ordinary server (See appendix **The NTLM Proxy module**).

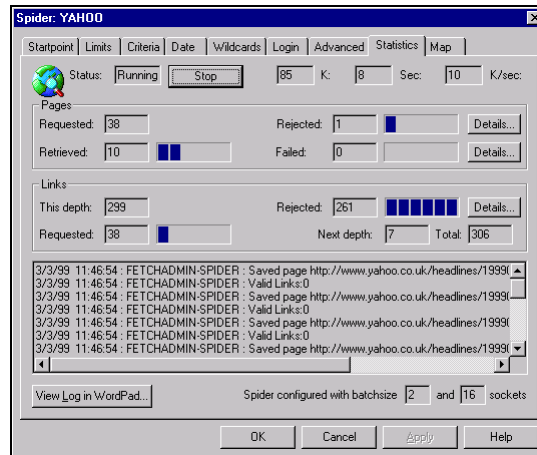
## SSL

### Use SSL Version

Whether or not you wish to use SSL. If you have selected to use SSL, this allows you to select the version of SSL you wish to use.

## The Site Statistics Page

The **Site Statistics** page allows you to test the spider parameters that you have specified in the previous set of pages. It allows you to run a test version of the spider that you have created and view various statistics on its performance. This enables you to fine-tune the parameters as well as detect any problems that might occur with the spider when running the fetch due to mis-configuration. To start the spider, press the **Start** button. The spider starts and the display on the **Start** button changes to **Stop**, which allows you to stop the spider before it finishes. The spider statistics will be displayed in the top portion of the window, and the tail of the log file is displayed in the lower portion of the window.



### Status

The status of the test spider can be **Running** or **Stopped**. To start the spider, press the **Start** button; the status display will change from **Stopped** to **Running** and the display on the button will change to **Stop** to enable you to stop the spider.

The next three displays show the **amount of data in Kilobytes** that has been downloaded during the current run of the spider, the **time in seconds** that the current run of the spider has taken so far and the **data transfer rate in Kilobytes per second** of the current run of the spider.

### Pages Requested

The number of pages that have been requested by the spider so far.

### Pages Retrieved

The number of pages that have been retrieved (i.e. downloaded) by the spider so far.

### Pages Rejected

The number of pages that have been rejected by the spider so far. Clicking the **Details** button displays the **Rejected Page Statistics** dialog, which contains the following details:

#### Rejected

The total number of pages rejected

#### Redirected

The number of pages redirected from one URL to another. This statistic is grayed out if you have set **Follow Redirection** to **on** in the **Start Point** page.

#### Links

The number of pages containing more links than specified in the **Page Criteria** page.

#### Size

The number of pages that fall outside the size range set in the **Page Criteria** page.

#### Can't Have

The number of pages that do not contain the **Must Have CSVs** in the page area specified to look in, set in the Wildcards Page.

#### Must Have

The number of pages that contain the **Can't Have CSVs** in the page area specified, set in the **Wildcards Page**.

#### Date Format

The number of pages that do not contain the date format in the page area specified, set in the **Date Page**.

#### Date Range

The number of pages whose date falls outside the range set in the **Date Page**.

### Pages Failed

The number of pages that have failed so far. Clicking the **Details** button displayed the **Failed Page Statistics** dialog, which contains the following details:

#### Pages Failed

The total number of pages failed.

#### Not Found

The number of pages not found i.e. URLs that return error 404.

#### Not Auth

The number of pages that the spider does not have authorization to access.

### **Failed (other)**

Any page failures that do not fall into the previous two categories. E.g. 500 Internal Server Error.

### **Links This Depth**

The number of links that have been found at this depth

### **Links Requested**

The number of links requested at this depth

### **Links Rejected**

The number of links that have been rejected so far. Clicking the **Details** button displayed the **Rejected Link Statistics** dialog, which contains the following details:

#### **Links Rejected**

The total number of links rejected

#### **Can't Have**

The number of links that contain the **Can't Have CSVs** (i.e. the can't have expression is contained within the URL) set in the **Wildcards Page**,

#### **Must Have**

The number of links that don't contain the **Must Have CSVs** (i.e. the must have expression is not contained within the URL) set in the **Wildcards Page**.

#### **Date Format**

The number of links that do not contain the date format (i.e. the date format is not contained within the URL) set in the **Date Page**.

#### **Date Range**

The number of links whose date falls outside the range set in the **Date Page**.

### **Links Next Depth**

The number of links at the next depth found so far.

### **Links Total**

The total number of links at this and the next depth found so far added together.

**View Log in Wordpad**

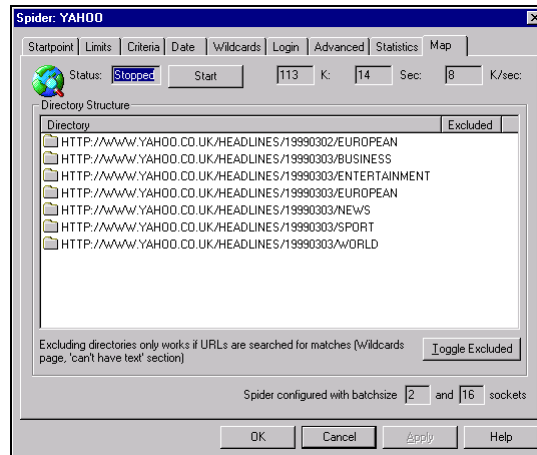
This opens up the log file for the spider so far in WordPad (text editor), so that you can examine the operation of the spider further.

**Spider Configured With Batchsize And Sockets**

This displays the batch size and number of sockets that the test spider is currently using.

## The Site Map Page

The **Site Map** page shows you a list of directory paths that are present in the site. It allows you to easily exclude and include (sets of) directories to spider. It does this by adding the directories you set to exclude to the **Can't Have CSVs** and setting the **Check in URL** flag, on the **Wildcards Page**. The **Start/Stop** button and status display is the same as that described on the Statistics Page.



### Status

This is described in the **Site Statistics** Page section.

### Directory Structure

This list is constructed as the spider runs and displays a list of all of the directory paths encountered so far. If you wish to exclude certain directories from the spider, select the directory(s) so that they are highlighted and click the **Toggle Excluded** button. The excluded directories will be indicated by a **Yes** in the **Excluded** column of the display. If you wish to re-include these directories, simply select them and click the **Toggle Excluded** button again. Included directories have an empty Excluded column entry.

### Options

This is described in the **Site Statistics** Page section.

### Spider Configured With Batchsize And Sockets

This is described in the **Site Statistics** Page section.





## 6. HTTPFetch Tutorial

---

### Spidering

#### Retrieving content from a remote web site

HTTPFetch uses a multi-socketed, parallel approach to retrieving content from a remote site. This allows spiders to pull down pages at a much higher rate than would be possible for a human.

It is important that you remember to be as “polite” as possible when spidering other web sites. Even if you have sufficient bandwidth available to use a large number of sockets (64 or more), the site you are spidering may not be able to handle that much traffic while still providing a reasonable level of service to other users. Additionally, large numbers of requests for pages coupled with no delay may appear to the site administrators to be a denial of service attack. It is advisable that you configure your spiders to be “polite,” or you may end up completely blocked from the site you wish to examine.

Here are three things you can do to avoid having your spiders blocked:

1. **Limit the number of sockets** – Set `NSOCKETS=` in the `HTTPFetch.cfg` file to a reasonable value, something on the order of 32 or less for *all* spiders. It will take longer to get all of your content, but you are less likely to encounter angry system administrators.
2. **Set a reasonable page delay** – Set `PAGEDELAY=` to a value from 1 to 5 seconds. This is the amount of time the spider will wait after it has retrieved a page before it requests the next link.
3. **Whenever possible, follow the Robots Protocol** – Set `FOLLOWROBOTSProtocol=` to 1. The Robots Protocol is designed specifically for sites to allow access to some pages, while restricting access to others. Allowed or disallowed pages are found in a file called `robots.txt` in the root directory of a site. The Robots Protocol is discussed in greater depth below.

## Following the Robots Protocol

The Robots Protocol is designed to let site owners allow or restrict access to specific areas of their site. A single file (per site) called `robots.txt` is placed in the root directory. Here is an example `robots.txt` file from <http://www.w3c.org/>, the web site for the World Wide Web Consortium:

```
# robots.txt for http://www.w3.org/
#
# $Id: robots.txt,v 1.17 1998/10/25 02:37:04 renaudb Exp $
#

# For use by search.w3.org
User-agent: W3Crobot/1
Disallow: /Out-Of-Date

# AltaVista Search
User-agent: AltaVista Intranet V2.0 W3C Webreq
Disallow: /Out-Of-Date

User-agent: *
Disallow: /Member/      # This is restricted to W3C Members only
Disallow: /member/      # This is restricted to W3C Members only
Disallow: /team/         # This is restricted to W3C Team only
Disallow: /TandS/Member # This is restricted to W3C Members only
Disallow: /TandS/Team   # This is restricted to W3C Team only
Disallow: /Project
Disallow: /Systems
Disallow: /Web
Disallow: /Team
Disallow: /History
Disallow: /Out-Of-Date
```

If HTTPFetch has been configured to follow the robots protocol, your spider will first retrieve `robots.txt`. Next, the spider will determine if there is a `User-agent` section which matches the value set in **SpiderAs** (see below). If there is such a section, the spider will follow the restrictions it specifies. Otherwise, it will look in the `User-agent: *` section, and follow the restrictions specified there.

The easiest way to determine if a site has restrictions on where it allows robots to go is to go to the root of the server in a browser and view `robots.txt`. For example, if you are planning to spider [www.cnn.com](http://www.cnn.com), you would go to <http://www.cnn.com/robots.txt> to view robot restrictions for the site.

If you find that the robots protocol definitions are too restrictive for you to get the content you're looking for, you may wish to set `FOLLOWROBOTS` to 0. If you do this, the spider will ignore `robots.txt` completely. It is a good idea to examine the site's `robots.txt` file anyway – you may be able to save yourself from downloading out-of-date or irrelevant pages by excluding certain directories (such as `cgi-bin`, etc.). If you do not have an arrangement with the site's owner, you will want to follow the other two “politeness” guidelines above (limit sockets, increase page delay). Additionally, you should take care to use the `SpiderAs` setting to send a reasonable user-agent string, such as `Mozilla/4.5 [en] (WinNT; U)` or `Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)`, so that your spider does not blatantly stand out in the remote site's logs.

## Using SpiderAs to identify yourself

All HTTP clients, including graphical browsers such as Microsoft Internet Explorer, Netscape Navigator, etc. and non-graphical browsers such as Lynx, Braille readers, etc. identify themselves to the HTTP servers they are connecting with using an environment variable called `HTTP_USER_AGENT`. The user-agent string includes information about the client's platform (such as Windows NT, Linux, MacOS, etc.), version, and language (English, Traditional Chinese, French, kanji, etc.). Web sites may be designed to serve different content to different user-agents. For example if the user-agent string indicates that the browser is MSIE 4.0 or greater, English language, serve a frames-based, ActiveX-enhanced version of the site, whereas if the browser is Lynx, Traditional Chinese, serve a non-graphical, double-byte character set version of the site. Additionally, many sites record the user-agent of incoming clients, so that they may determine which browsers are most popular and expend development effort on the site accordingly.

Autonomy's HTTPFetch allows you to configure the user-agent string the spider will send. By default, the spider will send the user-agent string "Autonomy Spider." Using the Fetch Administration module, you may use pre-defined user-agent strings for:

```
Microsoft Internet Explorer 4.0 (English)
Microsoft Internet Explorer 3.0 (English)
Netscape Navigator 4.5 (English)
Netscape Navigator 4.0 (English)
Netscape Navigator 3.0 (English)
Autonomy Spider
```

These values will allow you to spider for specific content as it would be served to those browsers. If you wish to use some other user-agent string or language, you must add the appropriate value to the `SpiderAs` setting in the `HTTPFetch.cfg` file. The `SpiderAs` value may be specified in either the `[DEFAULT]` section (to use the same value for all spiders), or in the individual `[SPIDERNAME]` sections (to use different values for one or more spiders).

If you would like to retrieve Navigator and/or IE-compatible content, but would still like to identify yourself in the log files of the site you are spidering, you may wish to define a custom user-agent string for `SpiderAs`. Using a string such as one of the following will give you the highest-capability pages:

```
Mozilla/4.0 [en] (compatible; MSIE 4.01; Autonomy Spider)
Mozilla/4.5 [en] (compatible, Autonomy Spider; I)
```

## Using SiteDuration to Finish Spidering Within an Allotted Amount of Time

You may use the `SiteDuration=` value to specify the amount of time (in seconds) you would like the spider to spend on a particular site. If you are using the Fetch Administration Module, you will find site duration under the **Limits** tab. `SiteDuration` may be set either globally (in the [DEFAULTS] section), or may be set on a per-spider basis. This setting will override site depth – if the spider has not reached the complete depth specified before the time limit expires, it will still stop and go no further.

For example, if you have arranged to spider a partner's site only at off-peak times (say, 12:00 AM until 2:00 AM), you might use the a `SpiderStartTime` (see ***Intelligent Scheduling***, below) of 00:01, and a `SiteDuration` of 7,140 seconds (1 hour 59 minutes) to complete your spidering within the time allowed.

## Using FollowRedirect

When a given page on a remote has been moved, the HTTP server sending the pages will respond with an HTTP header 302: Moved Temporarily, along with the URI for the page's present location. To the browser, this type of redirect is invisible – the browser is simply sent to the new page location. When spidering, you must decide how you wish to handle this situation. Setting `FollowRedirect=1` tells the spider that it should retrieve the new URL as if it were the original page, without counting the redirect as another level of depth. If you wish to ignore pages that have been moved, set `FollowRedirect=0`.

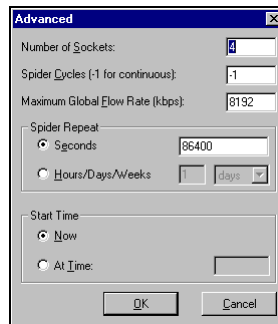
## Scheduling

### Intelligent Scheduling

A key to not wasting effort or bandwidth when running spiders is to be smart about how HTTPFetch is scheduled. If a site changes on a weekly basis, spidering it daily would not be productive. Alternately, if parts of a site change daily, but most of the site is static, it makes sense to only spider the parts of the site that change. Scheduling and repeats are handled in the following settings:

SpiderRepeatInterval	Time between spider cycles. Specified in human readable form e.g.hours. If this is not specified then SpiderRepeatSecs is used. <b>SpiderRepeatInterval will override SpiderRepeatSecs if both are specified</b>
SpiderRepeatSecs	This is the number of seconds before the spiders should repeat. This will be overridden by SpiderRepeat Interval.
SpiderCycles	Number of times the spider should attempt to index this site. A value of -1 indicates an infinite number of cycles.
SpiderStartTime	Clock time to begin the spider cycle, in 24 hour time. For example, a value of 23:15 would begin spider at 11:15 PM.

If you are using the Administration Module, open the **Advanced** window. You will see the following dialog:



**Note:** Scheduling occurs globally, not at the individual spider level. This means that you will need to set up multiple instances of HTTPFetch in order to spider at different intervals.

## Downloading Pages

### Including pages you want

There are two settings that control the pages to be included in your spider, `MustHaveCheck` and `MustHaveCSVs`. These values may be configured directly in the `HTTPFetch.cfg` file, or via the Fetch Administration Module. `MustHaveCheck` contains a bitmap defining where, and when the spider should look for matches. Possible values are:

1	<b>URL</b> Look for matches in the page's URL.
4	<b>Header</b> Look for matches within the <HEAD></HEAD> portion of the page.
8	<b>Body</b> Look for matches between the <BODY></BODY> tags of the page.
64	<b>Case Insensitive</b> Do matching regardless of case, e.g., <i>WORD</i> , <i>Word</i> , and <i>word</i> are all matches.
128	<b>Prevent Request</b> If there is no match, throw the page away e.g., discard the page, and do not look for further links within it.
512	<b>Valid site structure</b> The Fetch rechecks the <b>MustHaveCSVs</b> values for a page in order to make sure that the page is still valid before it updates it (if you don't make this setting, then changes to these values are never checked for). If the site is not valid, it is not downloaded.

Values may be combined to achieve the desired result. For example, if you want to check the page's URL, and only download the page if the URL contains something you're looking for, you would set a value of **129** (1 + 128).

The `MustHaveCSVs` parameter contains a comma-separated list of values, which the specified part of the page (URL, header, body, etc.) *must* contain in order to be spidered. The list may contain multi- or single-character wildcards, e.g.:

Word:	Matches:
*hello*	<a href="http://www.hello.com">www.hello.com</a>
	Hello, Fred
	Well hello there
?ell?*	Hello
	Jell-O

The values in `MustHaveCSVs` are evaluated using a logical **OR** operation. This means that if a page's URL, header or body (as specified above) match *ANY* of the values in the list, the page matches and it is indexed.

## Excluding pages you don't want

There are two settings that control the pages to be excluded from your spider, `CantHaveCheck` and `CantHaveCSVs`. These values may be configured directly in the `HTTPFetch.cfg` file, or via the Fetch Administration Module. `CantHaveCheck` contains a bitmap defining where, and when the spider should look for matches. Possible values are:

1	<b>URL</b> Look for matches in the page's URL.
4	<b>Header</b> Look for matches within the <HEAD></HEAD> portion of the page.
8	<b>Body</b> Look for matches between the <BODY></BODY> tags of the page.
64	<b>Case Insensitive</b> Do matching regardless of case, e.g., <i>WORD</i> , <i>Word</i> , and <i>word</i> are all matches.
128	<b>Prevent Request</b> If there is no match, throw the page away e.g., discard the page, and do not look for further links within it.

512	<p><b>Valid site structure</b></p> <p>The Fetch rechecks the <code>CantHaveCSVs</code> values for the site in order to make sure that the site is still valid before it updates it (if you don't make this setting, then changes to these values are never checked for). If the site is not valid, it is not downloaded.</p>
-----	--

Values may be combined to achieve the desired result. For example, if you want to check the page's URL, and exclude the page if the URL contains something you want to ignore, you would set a value of **129** (1 + 128).

The `CantHaveCSVs` parameter contains a comma-separated list of values which the specified part of the page (URL, header, body, etc.) *cannot* contain. The list may contain multi- or single-character wildcards, e.g.:

Word:	Matches:
*hello*	<a href="http://www.hello.com">www.hello.com</a>
	Hello, Fred
	Well hello there
?ell?*	Hello
	Jell-O

The values in `CantHaveCSVs` are evaluated using a logical **OR** operation. This means that if a page's URL, header or body (as specified above) match *ANY* of the values in the list, the page matches and it is indexed.

## The interaction between CantHave and MustHave values

Between the *Cant* and *Must* have lists, the spider performs a logical **AND** operation. This means that if the page you're looking at has something which is required (*MustHave*), but also has something which is forbidden (*CantHave*), it will be discarded.

## Limiting page retrieval with MaxPages

Another way to limit the amount of content you get back from a given site is to use the `MaxPages` setting (in the Administration Module under the **Limits** tab). `MaxPages` may be set either globally (in the [DEFAULTS] section), or may be set on a per-spider basis. Once the maximum number of pages has been reached, the spider will stop, regardless of depth (or any other consideration).



## Limiting page retrieval with TotalSize

Another way to limit the amount of content you retrieve is to set an upper limit on the amount (size) of the information you will retrieve. In the Administration Module, this is done in the **Limits** tab. It is accomplished using the `TotalSize` setting, which is specified in Gigabytes, Megabytes or Kilobytes.

`TotalSize=10gbytes`

`TotalSize=10000mbytes`

`TotalSize=10000000kbytes`

The total maximum is 2 TBytes (terabytes) which can only be set by using gbytes/mbytes/kbytes and not through the default bytes setting.

`TotalSize` specifies the amount of *textual* data, which will be downloaded to the system where the fetch is running. This is exclusive of graphics, multimedia content, and so on. Setting `TotalSize` to a large value may still result in an extremely large number of pages being downloaded.

To set the fetch to get more than 2Gbyte for a single spider, set the `TotalSize=10000mbytes` and it will fetch 10 GBytes, 10000000 kbytes will do the same.

## Limits interaction

The various limit types are evaluated using a logical **OR**. This means that if *any* of the limit criteria are met, the spider will stop retrieving pages until the next cycle.

## What page types should you exclude?

There are several types of pages you may wish to exclude when spidering sites. Here are a few things to look for.

Page Type	Why Exclude?	How to Exclude
Ad links	Content may be unrelated to what you are looking for (placed on the page for revenue rather than content purposes)	Identify ad server links (e.g., <code>adserv.myco.com</code> , <code>adcgi?parameters</code> ) and exclude using <code>CantHaveCSVs</code>  You can also set the value of <code>StayOnSite</code> to <code>TRUE</code> as most ad links are to external sites.
Link Pages	Pages containing link upon link do not generally contain much actual content. While they are good for finding additional links, they are probably not useful to index.	Use <code>MaxLinksPerPage</code> setting with a reasonable limit (30 to 50).
Off-Topic Content	If you are looking for information on politics, spidering information on football would be less than useful.	Use <code>CantHaveCSVs</code> and <code>MustHaveCSVs</code> to partition the site. For example:  <code>CantHaveCSVs=*/sport*</code> <code>MustHaveCSVs=*/politics/*</code>
Pages Linked To By Every Other Page	Many sites include links to site help, membership, copyright information, etc. on every page. Although identical URLs will only be indexed once, you may wish to exclude these pages, as they probably do not contain relevant content.	Use <code>CantHaveCSVs</code> to completely exclude the pages. You may include a complete path/page name to exclude that page, e.g., <code>*/help/about.html</code>

## Dealing with dates

The spider may be configured to include or discard pages based on a large number date criteria. There are two settings that control date settings, `DateCheck` and `DateFormats`. These values may be configured directly in the `HTTPFetch.cfg` file, or via the Fetch Administration Module. `DateCheck` contains a bitmap defining where and when the spider should look for date matches. Possible values are:

1	<b>URL</b> Look for date matches in the page's URL.
4	<b>Header</b> Look for date matches within the <HEAD></HEAD> portion of the page.
8	<b>Body</b> Look for date matches between the <BODY></BODY> tags of the page.
64	<b>Case Insensitive</b> Do matching regardless of case, e.g., <i>MARCH</i> , <i>March</i> , and <i>march</i> all match.
128	<b>Prevent Request</b> If there is no match, throw the page away e.g., discard the page, and do not look for further links within it.

The formats of the string that the spider should look for are configured in the `DateFormats` setting. `DateFormats` contains a comma-separated list of possible date strings. Please look at the 'Supported Date Formats' appendix for details.

Custom date formats may also be added.

## Restricting page retrieval with **AfterDate** and **BeforeDate**

In addition to examining each page for dates, you may also specify dates before and after which a given page will not be indexed. Generally, these two settings should be used in combination.

<b>AfterDate</b>	The page must have a matching date (as specified in <code>DateCheck/DateFormats</code> ) which is <b><i>after</i></b> the value specified. For example, <code>AfterDate=-7</code> would specify that the page date must not be more than seven days old.
<b>BeforeDate</b>	The page must have a matching date (as specified in <code>DateCheck/DateFormats</code> ) which is <b><i>before</i></b> the value specified. For example, <code>BeforeDate=1</code> would specify that the page date cannot be more than one day in the future.

## Importing

### Using ImportStripLinks

Normally, the text within an `<A HREF></A>` tag is not very descriptive of the current document – rather it is descriptive of the document that the hypertext link leads to. By default, HTTPFetch sets a parameter of `ImportStripLinks=1`. With this setting, the spider will ignore all text within hypertext links. If you would prefer to index the actual text of links, set this option to 0.

### Using fixed and variable fields

HTTPFetch spiders may be configured to add additional, custom information to each record as it is imported and prepared for indexing. Fixed fields are used to add the same, static information to each record. Variable fields are used to gather arbitrary information from each page (presuming that the information is formatted in a uniform way) so that it may be passed into each index record. Only modifying HTTPFetch.cfg may currently configure fixed and variable fields.

**NOTE:** In order to the information spidered into fixed or variable fields later, you must configure the DRE you will be indexing into to expect these fields, and to allocate enough storage for the information you are passing in. If you do not, the information in fixed and variable fields will simply be discarded at index time.

#### Example: Fixed Field

Fixed fields can be useful when you wish to associate a particular bit of information with the contents obtained by each spider. For example, you may wish to index in the name of a graphic that identifies content based on the location, from which it came, so that you can display the graphic at query time next to the result by using

`<!--ATNMY_DREFIELDBYNAMEname-->` in your templates. To do this, you could insert the following values into the spider definition for a given site:

```
FixedFieldName0=Image
```

```
FixedFieldValue0=my_image.gif
```

The contents of `FixedFieldValue` will be added to each imported record. To add additional fixed fields, simply begin with `FixedFieldName0` and `FixedFieldValue0`, and increment the number until all desired fixed fields have been added.

### Example: Variable Field

Variable fields can be extremely useful when the site either uses an extremely regular structure, which contains some information you are interested in having available as a field (for sorting or querying), or when you wish to override some default information the spider may retrieve. For example, many sites, particularly if they are being served dynamically, do not have extremely useful information within the <TITLE></TITLE> tags (which is where HTTPFetch will look by default for a record title). However, if the site contains a useful, distinguishing title with another set of tags on the page, for instance within <H2><FONT COLOR="#FFFFFF"></FONT></H2>, you can use a variable field to get a more descriptive title for each page. Defining a field, then defining the starting and ending strings that the spider should look for does this. Everything within the starting and ending strings will be included in the defined field. For example, you might add the following to the spider configuration for a given site:

```
FieldName0=DRETITLE
FieldStart0=<H2><FONT COLOR="#FFFFFF">
FieldStop0=</FONT></H2>
```

The spider will put the first instance it finds of the above pair into a field for you. Thus, a page containing the following:

```
<HTML>
<HEAD>
<TITLE>Generic Page Title</TITLE>
<BODY BGCOLOR="#000000">
<H2><FONT COLOR="#FFFFFF">Hand-Crafted Page Title</H2></FONT>
[...]
page content
</BODY>
</HTML>
```

would have the generic page title within the <TITLE></TITLE> tags overridden by a more descriptive title supplied within the body of the document.

### Fetching Meta Tag Data to Fields

Spiders may be configured to automatically import meta tag information into fields. To automatically add the contents of meta tags to fields, set `ImportMetaToFields=1` in `HTTPFetch.cfg`. This may be done at the global lever, or on a per-spider basis. Field definitions will be imported to a field of the same name as the meta tag, e.g., `<META NAME="description" CONTENT="Document description">` will become `#DREFIELD description="Document description"` when it is imported.

**You must be sure to create corresponding fields in the DRE configuration file before beginning your index job for anything useful to be done with this information.**

## Example Configuration

Following are example configurations of spiders for CNN (<http://www.cnn.com/>) and CNN Financial (<http://www.cnnfn.com/>). Some portions have been omitted. Note that some values are overridden in the spider definitions themselves.

```
// Defaults section
[DEFAULT]
NSOCKETS=24
SPIDERREPEATINTERVAL=86400
SPIDERCYCLES=-1
SPIDERSTARTDELETEOLD=on
LOG FILE=spider.log
DEPTH=3
SITEDURATION=3600
MAXPAGES=2000
FOLLOWREDIRECT=TRUE
STAYONSITE=TRUE
PAGEDELAY=0
MINPAGESIZE=1000
MAXPAGESIZE=1000000
MAXLINKSPERPAGE=30
PAGETIMEOUT=100
PAGEDURATION=600
EXTENSIONS=*.htm,*.html,*.shtml,*.fhtml,*.asp,*.cfm
DATECHECK=13
DATEFORMATS=LONGMONTH DD,LONGMONTH D,SHORTMONTH. DD,SHORTMONTH.
D,SHORTMONTH-DD,SHORTMONTH-D,SHORTMONTH DD,SHORTMONTH D,DD SHORTMONTH,D
SHORTMONTH,YYMMDD,YYMMDD,YYYY/MM/DD,MM/DD/YY,MM-DD-
YYYY,DDMMYY,MMDDYY,MMDDYY
AFTERDATE=-1
BEFOREDATE=10
IMPORTCHECKSUM=on
IMPORTSTRIPLINKS=on
IMPORTSUMMARY=on
IMPORTBREAKING=on
IMPORTMINLENGTH=100
IMPORTMAXLENGTH=999999
IMPORTMINLENGTHWORDS=100
IMPORTMAXLENGTHWORDS=999999
IMPORTBREAKINGMINPARAGRAPHWORDS=300
IMPORTBREAKINGMAXPARAGRAPHWORDS=600
IMPORTBREAKINGMINDOCWORDS=601
FOLLOWROBOTPROTOCOL=1

[CNNFN]
// Startpoint URL
URL=http://www.cnnfn.com
// Override stay on site setting
STAYONSITE=FALSE
```

```
// Page delay (politeness!)
PAGEDELAY=5
// Don't follow robots protocol, as CNN does not allow robots
FOLLOWROBOTPROTOCOL=0
// Check for match before downloading page
MUSTHAVECHECK=129
// Match requirements
MUSTHAVECSVS=*cnnfn.*
// Check URL, body for date
DATECHECK=9
// Use the following date formats
DATEFORMATS=/YYYY/MM/DD,/YMM/DD/,/MM/DD/YY/
// Send logging information to this file
LOG FILE=CNNFN.log
// Check URL before download
CANTHAVECHECK=129
// Can't match any of the following
CANTHAVECSVS=*SPORT*,*STYLE*,*SPEC*,*almanac*

[CNN]
// Startpoint URL
URL=http://www.cnn.com/WORLD/
// Override stay on site setting
STAYONSITE=FALSE
// Page delay (politeness!)
PAGEDELAY=5
// Don't follow robots protocol, as CNN does not allow robots
FOLLOWROBOTPROTOCOL=0
// Check for match before downloading page
MUSTHAVECHECK=129
// Match requirements
MUSTHAVECSVS=*cnn.*,*WORLD*,*/US*,*/ALLPOL*
// Check URL for date
DATECHECK=1
// Use the following date formats
DATEFORMATS=/YYYY/MM/DD,/YMM/DD/
// Send logging information to this file
LOG FILE=CNN.log
// Check URL before download
CANTHAVECHECK=129
// Check URL before download
CANTHAVECSVS=*SPORT*,*STYLE*,*SPEC*,*almanac*
// Override default depth
DEPTH=2
// Override default site duration
SITEDURATION=3600
// Additional URLs to spider
URL0=http://www.cnn.com/US/
URL1=http://www.cnn.com/ALLPOLITICS/
```



## Running the Fetch

### Fetch Directory Structures and Temporary Files

While it is running, HTTPFetch uses temporary files to keep track of the content it is looking at. For each configured spider, a subdirectory is created under the directory where you have installed HTTPFetch. This directory will have the same name as your spider. As the spider runs, it will create two kinds of temporary files – first, a file containing all links at this level will be created. Next, each page will be downloaded for analysis. If you wish to delete these files at each spider cycle (or whenever the spider is stopped and restarted), set `SpiderStartDeleteOld=1` in the [DEFAULTS] section of your `HTTPFetch.cfg` file.

### The Import (IDX) File Format

You may find on occasion that, although you have configured your spiders correctly and you are getting lots of content, the results you are getting are not what you expect. In this case, you may wish to examine spider's import files. Depending on the import mode you are using, you should file a `SpiderName.idx` file in the HTTPFetch directory. An `IDX` file will contain information that has been formatted in Autonomy's import format, so that it is ready to be indexed into the DRE. A typical `IDX` file entry will look like this:

```
#DRREFERENCE
http://www.mysite.com/us/DailyNews/teenviolence0325.html
#DRETITLE Why Do Teens Kill?
#DREFIELD Summary="Explaining Acts of Brutality by Youngsters Why Do
Teens Kill?"
#DREFIELD Image="MY_news.gif"
#DRESECTION 0
#DREDATE 925668001
#DREDBNAME Database
#DRESTORECONTENT y
#DRECONTENT
Explaining Acts of Brutality by Youngsters Why Do Teens Kill? Dr.
Howard Spivak,
  \chairman of AAP Task Force on Violence, comments on kids using
violence to
  \solve problems Luke Woodham is accused of killing his mother, then
going to
  \school and shooting nine students in Pearl, Miss.
  \[...]
  \content here
  \[...]
  \As a very common theme. In 80% of the cases it has been the case.
#DREENDDOC
```

Here is a point-by-point breakdown of the above IDX file entry.

Key	Value
#DREREFERENCE	Original URL for this page.
#DRETITLE	Document title. Generally obtained from the target page's <TITLE></TITLE> tags.
#DREFIELD <i>Summary</i>	Short summary of the document. Usually computer generated. Used when "Short Summary" option is selected in the CGI.
#DREFIELD <i>Name</i>	User-defined fixed and variable fields. If ImportMetaToFields is set to 1, there will be additional fields for META content from the page, such as <META NAME=description> and <META NAME=keywords>.
#DRESECTION <i>n</i>	If you are using document splitting, HTTPFetch will assign a sequential DRESECTION value for each section of the document.
#DREDATE <i>nnnnnnnnnn</i>	Document date in epoch seconds.
#DREDBNAME <i>DatabaseName</i>	Name of the database into which this content is to be indexed.
#DRESTORECONTENT <i>{y   n}</i>	Flag telling the DRE whether or not it should store this content. StoreContent=1 must also be set in the DRE.INI.
#DRECONTENT	#DRECONTENT is followed by a newline character, then the actual content to be indexed for this record. Content is bounded by a newline character, or by the next #DRE key encountered.
#DREENDDOC	Indicates the end of this record.

## Log Files

HTTPFetch creates two kinds of log files. The first log file is a general, overall HTTPFetch log. Additionally, spiders log their activities to one or more log files. By default, the overall log will be *InstallationName.log*, while the default spider log is (aptly), *spider.log*.

Entries in the HTTPFetch.log file will vary, depending on whether or not you are running in batch mode, and if you are using batch import or batch index. Here is a short snippet from a successful run of the HTTPFetch:

```
3/5/99 20:18:47:----- Import: Importing MRFEATURES to
D:\Autonomy\ABC_HTTPFetch\MRFEATURES.idx...

3/5/99 20:18:47:Import: Success

3/5/99 20:18:47:----- Import: Importing MRPEOPLE to
D:\Autonomy\ABC_HTTPFetch\MRPEOPLE.idx...

3/5/99 20:18:47:Import: Success

3/5/99 20:18:52:----- Index: Indexing
'D:\Autonomy\ABC_HTTPFetch\ABCNEWS.idx' as
'D:\Autonomy\ABC_HTTPFetch\ABCNEWS.idx'

3/5/99 20:18:52: -> DRE on '127.0.0.1:13051'...

3/5/99 20:18:52:Success
```

Each line begins with a date and time stamp. The time stamp format is D/M/YY HH:MM:SS. Next, the operation that is being performed is listed (e.g., Import or Index). This is in turn followed by a status (e.g., **Success** or **Failure**). If a failure is encountered, the reason for the failure will be listed in the log file.

Each spider will also log its activity to a file. It is generally a good idea to configure each spider with its own log file, so that you may more easily determine what the spider has been doing. Here is a short snippet from a successful spider log file:

```
30/4/99 10:00:21 : MRFEATURES : Saved page
http://mrshowbiz.go.com/features/oscars_99/photo19.html.

30/4/99 10:00:21 : MRFEATURES : Getting link
http://mrshowbiz.go.com/features/oscars_99/photo20.html

30/4/99 10:00:21 : MRFEATURES : Valid Links:0

30/4/99 10:00:21 : MRFEATURES : Saved page
http://mrshowbiz.go.com/features/oscars_99/photo20.html.

30/4/99 10:00:21 : MRFEATURES : Getting link
http://mrshowbiz.go.com/features/oscars_99/photo21.html

30/4/99 10:00:22 : MRFEATURES : Valid Links:0

30/4/99 10:00:22 : MRFEATURES : Saved page
http://mrshowbiz.go.com/features/oscars_99/photo21.html.

30/4/99 10:00:22 : MRFEATURES : No more links.
```

## HTTPFetch Tutorial

30/4/99 21:10:16 : MRFEATURES :

-----

### Statistics:

Total Bytes:	14699195
Total Duration:	3021s
Total Bytes/Sec:	4865
Pages Analysed:	1006
Pages Saved:	1005
Redirected:	0
NotFound:	0
Unauthorized:	0
Failed(?):	0
Total:	0

### Pages Rejected:

Size:	8
Links:	0
CantHave:	0
MustHave:	0
DateFormat:	0
Date:	0
Total:	8

-----

Links Analysed: 52508

Requested:	31479
------------	-------

### Links Rejected:

CantHave:	0
MustHave:	13720
DateFormat:	0
Date:	0
Total:	13720

As with the `HTTPFetch.log`, each entry in the spider log includes a time stamp. When a page is requested, the spider will attempt to download it. If a server error is sent back by the remote site (e.g., 404 - Not Found), that information will be written to the log file. Additionally, the spider provides statistics about the pages it visited, links it accepted or rejected, and the reasons for each. If you find that you are not getting all the content you expected, or you are getting too much, look at the statistics section of the spider log. You may find that your `CantHave` or `MustHave` parameters have been too permissive, or that the site you are spidering has invalid links, or that other errors have occurred. The following list of HTTP error codes will be of assistance in interpreting the spider log.

Server Response:	Meaning:
200: OK	Normal page response for a successful page request.
302: Temporarily Moved	The page you requested has been moved to another location on the server. Normally, this is not seen by a browser, which is automatically sent to the new page location. The <code>FollowRedirect</code> setting in <code>HTTPFetch.cfg</code> determines how the spider will deal with this response.
404: Not Found	The page you requested does not exist, or the server has been told to deny you access to it.
500: Server Error	The CGI you have called has responded with an error.

## Manipulating HTML Rendered files

The OmniSlave and PDF slaves temporarily convert all the various formats (i.e. Word, PDF, Excel, PPT, etc.) into HTML. This temporary rendered HTML is then processed to create data in IDX ASCII format. If you wish, you can store the temporary HTML data on a repository on the web server so that when you serve links to results you can also serve links to the rendered HTML.

In order to implement this, the following steps are required internally:

1. After conversion, copy the rendered HTML to a hashed directory structure on the web server
2. Set a DREFIELD to point to the location of the rendered HTML. Reference conversion might be needed as in standard document references.

The import parameters needed to set this up in the indexing product are:

**ImportRenderedHTMLMoveToDir**

Directory where the rendered HTML should be moved to.

**ImportRenderedHTMLExtensions**

The extensions of documents for which the rendered HTML should be kept.

**ImportRenderedHTMLFieldName**

The name of the DREFIELD where the path of the rendered HTML is kept.

In addition to these, you can perform any reference replacement necessary to get the right path. These settings are equivalent to the set used for the DRE document reference.

ImportRenderedHTMLPathReplaceUpToSlash

ImportRenderedHTMLPathReplaceString

ImportRenderedHTMLRefReplaceCSVs

ImportRenderedHTMLRefReplaceWithCSVs

## 7. HTTPFetch Troubleshooting

---

### Why is the HTTPFetch not fetching any data?

There are several reasons why the HTTPFetch might not be able to download pages from the external web sites. The best way to ascertain why data was not retrieved is to check the LOG file and read through the error messages.

Some of the common reasons are:

- The server does not have external HTTP access: make sure the server has the appropriate connections available.
- Robots protocol does not allow it: check the file /robots.txt and make sure that for example “/” is not there. If “/” is present, then it means that the site does not allow any spidering.
- Some sites only allow certain spiders to enter their site: you should check the /robots.txt for the value of “User-agent”, if it has the value “\*” then the site should allow any spiders, otherwise it will list spiders that are allowed on the site. You can pretend to be a different spider by setting the “SpiderAs” value in the .CFG file.
- The spidered Web Sites are down: if the web sites are not live, then there is not much that the Fetch can do.
- Some web sites don't allow more than N connections per site: if for example you have one site and 32 sockets set in your Fetch, and the site only accepts 10 simultaneous connections from one server, you will miss data on the connections that don't manage to get through.
- Multiple URLs per spider set without *StayOnSite*=false: when you set more than one URL in a spider, the value of URL= is the principal one. If the values of URL1, URL2, ...etc. are not from the same web site and you have set *StayOnSite*=true, the Fetch will not spider all of them successfully.

### Why has the HTTPFetch not managed to index the data?

If the Fetch has finished a cycle and the data doesn't seem to have been indexed, there could be various reasons for this:

- There was no data to index: check the configuration of the spiders, and check if any pages were downloaded.
- The DRE was not running: check that the DRE is up and running correctly.
- The Fetch was not configured to index the right DRE: check the Fetch was pointing to the right DRE.

## Why does the HTTPFetch get so many uninteresting pages?

In all data management systems, the rule of “garbage in, garbage out” always applies. It is very important that the HTTPFetch is configured as precisely as possible in order to get meaningful data indexed into the engine.

You should always look at the web site you are going to spider and determine at least the following things.

- Decide which sections of the web sites you want to spider (e.g. news, entertainment, law reports, etc..). To set this up in the Fetch you need to set the *MustHaveCSVs* value in the .CFG file.
- Decide which sections of the web sites you definitely DO NOT want to spider (e.g. archive, opinions, shopping, etc..). To set this up in the Fetch you need to set the *CanHaveCSVs* value in the .CFG file.
- Decide what date range you want the information to be in (e.g. 1 day, 3 days, 1 week, etc..). You can set the date range in the .CFG file by using a combination of the values *DateFormats*, *AfterDate* and *BeforeDate*
- Decide whether you want to download headline pages or not. The value *MaxLinksPerPage* is used for this in the .CFG file.
- Determine how much data you want to obtain: maximum number of pages, maximum depth, etc..

## Why does the HTTPFetch stop spidering half way through?

If you suddenly find that the Fetch stops, the log file should tell you why this happened. Some of the common reasons are:

- *SiteDuration* time was reached
- *MaxPages* value was reached

## Why are documents not being imported/indexed?

Probable cause: the document is outside the document size criteria. Check the *ImportMinLength*, *ImportMaxLength*, *ImportMinLengthWords* and *ImportMaxLengthWords* settings. If these appear to be correct for the documents in question, enable import logging to see why the document is not being imported. The settings required are

`ImportLogFile=import.log`

`ImportLogFileAppend=true`

`ImportLogLevel=FULL.`

This will produce a file, `import.log`, containing information about why documents are not being imported which will help track down the problem.



## Why is my spider downloading so many pages each time it runs?

The spider stores the structure of a site, and calculates when to revisit pages based on the history of modifications to the pages. This reduces bandwidth, disk, and processor time. It should always grab everything to begin with and then less and less with time, until only the changing pages are re-grabbed.

Use:

```
StoreSiteStructure=true
```

to enable this.

Once enabled you can also add:

```
DeleteProcess=DREREMOVE
```

which makes the spider automatically remove pages from the DRE that have gone 404.

## Why does the HTTPFetch fail to access certain web sites?

If a web site requires authentication, use the **Edit Spider's Login** page to provide the HTTPFetch with the login information that this site requires.

If the HTTPFetch fails to access a site although you have provided the extra login information, the site may require NTLM authentication. In this case, the HTTPFetch can only access this site if you have installed and configured the NTLM Proxy module. See appendix **The NTLM Proxy module** for details on how to find out whether a web site requires NTLM authentication, how to configure the HTTPFetch to point to the NTLM Proxy module and how to configure the NTLM Proxy module itself.



## 8. HTTPFetch FAQ

---

### How fast is the HTTPFetch?

The speed of the HTTPFetch depends on a series of factors:

- Bandwidth of the connection: the bigger the bandwidth the faster the process of downloading data.
- Power of the server (i.e. free disk space available, memory available, processor power, etc.) relative to how much data the HTTPFetch is dealing with.

### Can the HTTPFetch get stuck in an infinite loop?

No, the spidering process of the HTTPFetch is done in breadth first mode rather than depth first mode. This means that when traversing the web site tree the spiders do not go down one branch until they hit the end. The spiders always process all links up to the depth specified, and then pages are retrieved in order of depth.

### What do the error messages in the log file mean?

It is very important for the administrator to understand the error messages shown in the HTTPFetch log file. It is advisable to specify a separate log file for each spider. The log file gives guidelines on why the data was not fetched successfully in order to adjust the configuration of the HTTPFetch.

Below is a list of the most common messages in the log file:

CATEGORY	MESSAGE IN LOG FILE	DESCRIPTION
PAGE CRITERIA	Page Criteria: Too many links.	The page had too many links in it and hence it was not saved.  If you would like, you can change the Value of <i>MaxLinksPerPage</i> in the .CFG file.
PAGE CRITERIA	Page Criteria: Too small.	The page was too small and hence it was not saved.  If you would like, you can change the Value of <i>MinPageSize</i> in the .CFG file.
PAGE CRITERIA	Page Criteria: Contains 'CantHave' expressions in body.	The page contained one or more of the <i>CantHave</i> expressions listed in the .CFG file.
PAGE CRITERIA	URL http://www.bot.or.th/govnr/public/loi/loi-thai/loi5/Box-E.doc didn't match extensions list *.htm, *.html, *.shtml, *.fhtml, *.asp' - not saved.	The URL was not in the list of the allowable extensions

CATEGORY	MESSAGE IN LOG FILE	DESCRIPTION
DATE CRITERIA	<URL> last modified date out of range.	The Date Criteria did not match. In this case the date information was found in the body of the page.
DATE CRITERIA	Page Criteria: Date out of range in body.	The Date Criteria did not match. In this case the date information was found in the body of the page.
HTTP CONNECTION	<URL>: Not Found (404).	The page that the spider tried to get could not be found.
HTTP CONNECTION	Unhandled: Forbidden (403).	The web site did not give permission to spider that page.
HTTP CONNECTION	Unhandled: Internal Server Error (500).	The web site gave an internal server error.
HTTP CONNECTION	<URL1> redirected to <URL2> : Moved Temporarily (302).	The spider was redirected to another URL. If you have <i>FollowRedirect</i> set in the .CFG file then the spider should follow the redirection and carry on.
HTTP CONNECTION	Content length discrepancy for <URL>  Expected length = <LENGTH1>, Actual Length = <LENGTH2>.	The spider could not receive the data in its entirety. This could be due to the actual web site, or the connection between the two servers.
HTTP CONNECTION	Retrying URL <URL> for the 0 time – Couldn't recv.	The spider could not receive data from the web site, spiders will retry up to 3 times before giving up.
HTTP CONNECTION	Retrying URL <URL> for the 0 time - Couldn't connect.	The spider could not connect to the web site, spiders will retry up to 3 times before giving up.
HTTP CONNECTION	Exceeded page timeout on <URL>	The page took too long to download. If you want the spider to wait longer modify the <i>PageTimeout</i> value in the .CFG file.
HTTP CONNECTION	internal HTTP://10.1.1.122/internaltestdata /:Unauthorized (401).	<p>You are trying to access a web site that requires authentication. In order to be able to fetch from this site, use the <b>Edit Spider's Login</b> page to provide the HTTPFetch with the login information that this site requires.</p> <p>If the HTTPFetch fails to access a site although you have provided the extra login information, the site may require NTLM authentication. In this case, the HTTPFetch can only access this site if you have installed and configured the NTLM Proxy module (See appendix <b>The NTLM Proxy module</b>).</p>

CATEGORY	MESSAGE IN LOG FILE	DESCRIPTION
SPIDER END	Finished: Maximum site duration reached.	The maximum site duration was reached.  If you would like the HTTPFetch to run for longer then please modify the <i>SiteDuration</i> value in the .CFG file.
SPIDER END	No more links	No more links were found to follow and hence the spider finishes.
DATA TYPE	Binary File	The link obtained was to a binary file.
SITE NAVIGATION	Robot protocol disallowed: <URL WILDCARD EXPRESSION>	It indicates which sections of the web site were not spidered due to restrictions from the robots protocol.  If you would like the spiders not to follow the robots protocol then please modify the <i>FollowRobotsProtocol</i> value in the .CFG file.
IMPORT	Import: Importing <SPIDERNAME> to <NAME>.idx  Import: Success	Shows whether the data importing was successful for a particular spider.  If it was not successful, you should check whether the spider did in fact download some pages and adjust the configuration if necessary.  Also check that the <i>ImportPath</i> setting is correctly set in the .CFG file.
INDEX	Index: Indexing <FILENAME>.idx' as <FILENAME>.idx'  -> DRE on '<HOST>:<PORT>'  Success	Shows whether the indexing of data was successful.  If it was not successful, you should check that the Fetch is configured to index into the right DRE and that the DRE is running properly.  You should also check whether the import stage was successful.  Also check that the <i>IndexPath</i> setting is correctly set in the .CFG file.

## **What happens if I connect to the same page twice, referring to it once by the Fully Qualified Domain Name, and then again by IP address?**

These will be classed as two separate pages so INDEXMODE will have to be used in this situation (MATCHnn) so only one of the documents will be indexed

## **Why do I get a message that the HTTP header is returned?**

This is not an error, it is a warning. The web server probably has not returned a date of any sort, this warning is there to say that you cannot rely on the last modified date being accurate when it is indexed into the DRE. The fetch is working fine, the page has been downloaded, the links from it will still be extracted etc, it is just that the DREDATE may not be the last modified date.

## **Can I spider dynamic web pages?**

Yes, you can spider dynamic web pages. A dynamic page is no different from a static one by the time it gets to the browser/spider. It has to be basic HTML by definition otherwise the browser won't be able to display it. As long as the same URL results in the same page on every request then the URL stored by the spider (and the DRE) will correctly identify the content received by the spider and therefore there is no problem. There are a number of things that can suggest that there may be a problem. To avoid these, ensure that the following is true:

1. Use a page delay. If you do not, the remote site's dynamic generation tool may not keep up with the spider. Some sites can get overloaded relatively easily so it is best to limit the number of sockets used and put a page delay in.
2. Sometimes forms are used to initiate navigation (like search forms, database inquiry forms). These forms need to be submitted. Spiders DO NOT submit forms automatically. This is to prevent chaos from spiders registering millions of times etc. on e-commerce sites or purchasing a large number of items. In some cases you can use a GET request (an assembled URL) to submit a form instead.
3. Check that the URL doesn't expire for a given page. An easy check is to add a URL for a dynamic page to your favorites and check it the next day. If the same content is returned all is well.
4. If JavaScript is used to generate the hyperlinks that a user navigates through it may be necessary to use the NavLinksToFollow parameters to extract links between things such as "open.window(" and ");", as there may be no genuine <HREF= links.
5. Avoid areas that generate millions of pages of no value (for example pages of purely numerical data from a database query, when there are potentially millions of combinations). This is easy with the Nav dir/site defs and must/cant have.
6. Avoid areas of sites where you can end up at the same page repeatedly with a different URL each time.

## How do I spider sites that start with an HTML FORM?

1. Go to the page containing the form
2. View the source of the page (or specific FRAME) containing the form.
3. Determine the form ACTION attribute. Amalgamate this with the current URL if it is relative or just take the whole thing if it is absolute. It should look like:  

```
<FORM ACTION="http://www.myserver.com/mycgi.exe">
```

 so you take the:  

```
http://www.myserver.com/mycgi.exe
```

 or  

```
<FORM ACTION="/cgi-bin/mycgi.exe">
```

 so you take:  

```
http://www.myserver.com/cgi-bin/mycgi.exe
```

 assuming the original URL was `http://www.myserver.com/form.html`
4. Next look for all the <INPUT tags on the page. Each should have a name and value attribute:  

```
<INPUT TYPE=HIDDEN NAME='MyFirstVariable' VALUE=helloworld>
```

```
<INPUT TYPE=TEXT NAME=Option VALUE="56">
```

 Write down all the name:value pairs.
5. Next make a full URL from the Action URL you extracted and all of the name value pairs:  

```
http://www.myserver.com/mycgi.exe?Option=56&MyFirstVariable=helloworld
```
6. Try the full URL in a browser and you should get the same results as if having posted the form. If you do then you can use this technique. If not, you will need to perform a POST submit using the Login Post features.
7. Should the above method work then take your URL and paste it into the fetch cfg:  

```
URL=http://www.myserver.com/mycgi.exe?Option=56&MyFirstVariable=helloworld
```
8. Add any further combinations of variables you require to submit different values for the form:  

```
URL=http://www.myserver.com/mycgi.exe?Option=56&MyFirstVariable=helloworld
```

```
URL0=http://www.myserver.com/mycgi.exe?Option=56&MyFirstVariable=helloworld
```

```
URL1=http://www.myserver.com/mycgi.exe?Option=56&MyFirstVariable=helloworld
```

```
URL2=http://www.myserver.com/mycgi.exe?Option=56&MyFirstVariable=helloworld
```

## **How do I remove documents from the DRE that have been removed from the site being spidered?**

If a site is returning pages with a 404 error message you may find that the pages have expired. You may find that these pages are still stored in the DRE, to overcome this problem when you have StoreSiteStructure=on you can then set DeleteProcess=DREREMOVE, which makes the spider automatically remove pages from the DRE that have gone 404.

## **What does the Date Page do and how do I know which setting to use?**

The use of URL, Header and Content in date checking to get the most current documents but not excluding yesterdays documents that have not been posted on a site until today the best thing to do is to look at the page itself. This will help determine which area of a page should be used to define the date process. In some sites, the URL itself will contain a date from which you can determine whether or not the page should be processed. The header information (which you can see by viewing the source of the page once you've loaded it into your browser) can also contain date information, as can the text content of the page itself. You can use an individual value or a combination of the three, also specifying in the same dialog whether, for example, you want to download the page first and then check the date(s), ensure the date falls in a given range of days, and so on.

## **What should I do with multiple index and content pages**

To overcome the problem of multiple index pages you may want to set up additional startpoints for your spider or indeed separate spiders. Each startpoint will start on the page with the highest number of links possible and each spider will be set to populate the same database in the DRE.

## **What is the Default Extension on the Import GUI Input Page and what makes it different from the extensions above it?**

When the spidering process encounters a file that doesn't have a "normal" HTML extension, such as .htm or .html, this value dictates how Autonomy is to treat the file. Leaving the Default value blank tells Autonomy to ignore files of unknown types (the supported types are described in the Import Parameters appendix). If you were to set the Default value to HTM it would tell the spider to treat the unknown file as it would an HTML file.



### **What does checking "Recurse down directory tree do? And what is the "Temporary Directory" box for?**

Directory recursion instructs the spidering process to "chase" the directory tree, automatically processing all files that are located hierarchically in and below the directory in question. The Temporary Directory exists in the event that vast amounts of data sets are being processed and the normal amount of working directory space used by the spidering process is exceeded (e.g. the local file system reaches capacity).

### **What is the highest setting for bytes and words to be imported?**

The highest value for words is 2,147,483,647.

The highest value for bytes is 1,874,919,423.

### **What is the difference between this date set in the import parameters and the DateCheck from the fetch setting?**

The earlier date specification is used to drive the selection or rejection of the page. If, having selected a page for processing, you wish to store a date from the page source into the DREDATE field or a locally defined field (as opposed to the date of the spider's execution) it is set up in the Data tab of the DRE Admin interface.

### **What is rendering?**

When Autonomy performs fetch processing the various formats of documents (Word, PDF, PPT, etc.) are converted--rendered--to temporary HTML files which are then used for indexing. If needed, the rendered HTML data can be stored for delivery to a web server, allowing you to serve links to the actual results as well as the rendered HTML.

### **What happens if we don't filter binary content?**

Binary files are not normally included in fetch processing. There are esoteric applications, however, that store meaningful text as part of a binary file, and the parameters on the advanced tab of the DRE Admin interface allow you to include such files in fetch processing, delimiting the values that dictate how the text in the binary image is to be defined.

## How does the spider DateCheck parameter work?

If you tell it to search the response header, the URL, the page header and page body:

DateCheck=0+1+4+8  
i.e. DateCheck=13

The spider will use the FIRST date found in a suitable format in the order searched, which is:

HTTP HEADER (0) URL (1) , <HEAD> (4), <BODY> (8)

and then it won't look for further dates...

Failing to find any suitable date anywhere means it uses the CONTENT date from the HTTP header (instead of the MODIFIED Date, that wasn't found) and failing a suitable CONTENT date it will use the current time (spider time).

## What parameters are required in order to remove old documents from the DRE when using the HTTPFetch?

Specify the following:

StoreSiteStructure=true

DeleteProcess=DREREMOVE

With the HTTPFetch with StoreSiteStructure on the product takes a snap shot of the links that have been followed. When it re-runs using the snap shot it can detect if things have changed or removed. If it comes across a page that has been removed, a 404 error is generated "Page not found". If this is one of the links previously spidered, the product send a DREREMOVE command to the DRE and the related document is removed.

If the \*.status files are deleted from the HTTPFetch directory the structure will be lost and documents will not be removed.

**Note:** this only works if the link has not been removed as well as the page.

## Appendix A: Service settings for Autonomy DiSH

---

HTTPFetch behaves as a standard Autonomy Service and can be used in conjunction with DiSH.

If the following settings are added to the configuration file, the service port is enabled and will accept the standard status and control commands.

[Service]	Description	Default
SERVICEPORT	DiSH listens for commands on this port	None
SERVICECONTROLCLIENTS	This specifies those who have control by wildcard IP match	None
SERVICESTATUSCLIENTS	This specifies those who can get the status of services by wildcard IP match	None

Standard Status and Control commands on the service port:

ACTION	Control/ Status	DESCRIPTION
GETSTATUS	STATUS	This is used to find out if a service is running
GETSTATUSINFO	STATUS	This is used to find out the product name, version etc.
GETCONFIG	STATUS	This is used to obtain the configuration file
SETCONFIG	CONTROL	This is used to set the configuration file
GETSTATISTICS	STATUS	This is used to gather statistics from the service
START	CONTROL	This is used to start a service
STOP	CONTROL	This is used to stop a service
PAUSE	CONTROL	This is used to pause a service
MERGECONFIG	CONTROL	Merge the fragment of the configuration file passed with the services configuration file.



## Appendix B: The NTLM Proxy module

---

If a site uses NTLM authentication you need to install the NTLM Proxy module in order to enable HTTPFetch to access this site. Note that while HTTPFetch can run on Windows NT, Solaris, Linux or HP-UX10, the NTLM Proxy module can only be installed on Windows NT (as it relies on NT DLLs).

### Installing the NTLM Proxy module

To install under Windows insert the NTLM Proxy module CD-ROM into your CD-ROM Drive. If your Windows installation is configured to support it, inserting the CD-ROM will automatically start the NTLM Proxy module installation program. Otherwise you can start the installation by double-clicking on the NTLMProxy2.2.x.EXE program in the root directory of the CD-ROM through Windows explorer.

Read and follow all installation instructions on the screen carefully. Before the installation program can start to copy files onto your PC, you need to provide it with some information:

1. The installation opens with the **Welcome** dialog. Read the text and click on **Next**.
2. The **Autonomy NTLM Proxy License** dialog is displayed. Read the license agreement and click on **Next** to accept it.
3. The **Installation Name** dialog is displayed.  
Enter a unique name for the NTLM Proxy installation, and click on **Next**. Note that the unique name must not contain any spaces.
4. The **Choose Destination Location** dialog is displayed.  
Select the directory in which you want to install the NTLM Proxy, and click on **Next**. By default DiSH is installed on **C:\Autonomy\NTLMProxy**, but you can use the **Browse** button to navigate to another location.
5. The **Service Settings** dialog is displayed.  
Enter the following details, and click on **Next**:
  - Port Number**  
The port by which requests are sent to the NTLM Proxy.
  - Number of threads**  
The number of requests that the NTLM Proxy can process at any one time.
6. The **Services** dialog is displayed.  
Indicate if you want the NTLM Proxy to start up immediately after the installation is completed, and click on **Next**.
7. The **Start Menu Shortcuts** dialog is displayed.  
Indicate whether you want to add shortcuts to the Windows Start menu and click on **Next**.

8. The **Start Installation** dialog is displayed.

Click on **Next** to confirm the settings you have made and start the installation. Alternatively click on **Back** to return to previous dialogs and make the appropriate changes.

9. The **Installing** dialog is displayed.

The progress of the installation process is indicated. If you want to abort the installation process, click on **Cancel**.

10. The **Installation Complete** dialog is displayed. The NTLM Proxy has been installed successfully. Click on **Finish** to exit the installation.

## Finding out whether a site uses NTLM authentication

If HTTPFetch fails to fetch a site due to authentication being required, contact your systems administrator to find out if the site uses NTLM authentication. Alternatively, use Netscape to browse the site. If you cannot log on to the site using Netscape it indicates that the site may use NTLM authentication.

## Pointing the HTTPFetch to the NTLM Proxy module

You can point the HTTPFetch to the NTLM Proxy module using one of the following ways:

### Using the Edit Spider dialog's Advanced page

1. Double-click on the **HTTPFetchAdmin.exe** to display the **Autonomy Fetch Administration Utility**.
2. Select the spider that you want to use the NTLM Proxy module, and click on the **Edit** button. The **Edit Spider** dialog is displayed.
3. Select the **Advanced** page, and use the **Proxy settings** to point the HTTPFetch at the NTLM Proxy, so that it can feed back the site to the HTTPFetch:

#### Use a Proxy Server

Check this box to use the NTLM Proxy module.

#### Host

Enter the address of the NTLM Proxy module.

#### Port

Enter the port that the NTLM Proxy module uses.

#### Username

Enter the name of the user who will access the NTLM authenticated site via the NTLM Proxy module.

#### Password

Enter the password that will allow the specified user to access the site.

4. Click on **OK**. The dialog closes, and the HTTPFetch is now pointing to the NTLM Proxy module.

### Using the HTTPFetch's configuration file

1. Open the HTTPFetch configuration file.
2. Scroll to the following parameters, and use them to point the HTTPFetch at the NTLM Proxy, so that it can feed back the site to the HTTPFetch:

#### ProxyHost

Enter the address of the NTLM Proxy module.

#### ProxyPort

Enter the port that the NTLM Proxy module uses.

**ProxyUsername**

Enter the name of the user who will access the NTLM authenticated site via the NTLM Proxy module.

**ProxyPassword**

Enter the password that will allow the specified user to access the site.

3. Save the configuration file and close it. The HTTPFetch is now pointing to the NTLM Proxy module

## Configuring the NTLM Proxy module

The configuration file of the NTLM Proxy module comprises the following default section, from which the required parameters are read.

**[default]**

**numthreads**

Enter the number of threads that the NTLM Proxy should contain. The maximum number of threads is 32.

**port**

Enter the NTLM Proxy module's port number.

**Note:** When the HTTPFetch tries to access a NTLM authenticated web site, it sends a request to the NTLM Proxy module rather than the web site itself. The NTLM Proxy module can only deal with one request at a time, since it cannot spawn more than one thread per request. We therefore recommend that you don't set the number of sockets in the HTTPFetch to be higher than the number of threads in the NTLM Proxy module.



## Appendix C: Supported date formats for spidering

---

These format specifiers are used whenever dates can be specified in the DDMMYY format style strings.

Format specifiers	Explanation
YY	This is used for the year when it is specified using 2 digits, for example when the year 1999 is specified as 99, 2000 as 00 and so on.
YYYY	This is used for the year when it is specified using 4 digits.), for example. 1999, 2000, 2001 etc.
LONGMONTH	This is used when the month is specified by the full name of that month, for example, January, March, August etc.
SHORTMONTH	This is used when the name of the month is reduced to an abbreviation, for example Jan, Mar, Aug etc
MM	This is used when the month is represented by a two digit number, for example 01,10,12 etc.
M+	This is used when the month is represented by a number which may consist of either one or two digits, for example 1, 2, 3, 10 etc.
DD	This is used for the day when it is represented by a two digit number, for example 01,02,03,12,23 etc.
D+	This is used for the day when it is represented by a number which may consist of either one or two digits, for example 1,2,12,13,31 etc.
HH	This is used for the hour when it is represented by a number which consists of two digits, for example 01, 12, 13 etc.
H+	This is used for the hour when it is represented by a number which may consist of either one or two digits.
NN	This is used for the minute when it is represented by a number consisting of two digits.
N+	This is used for the minute when it is represented by a number which may consist of either one or two digits.
SS	This is used for the number of seconds when this is represented by a two digit number.
S+	This is used for the number of seconds when this is represented by a number which may consist of either one or two digits.
ZZZ	This is used for the time zone abbreviation, for example, GMT, EST, PST etc.

**Examples:**

For dates with days which are represented by either one or two digits:

DateFormats=D+/SHORTMONTH/YYYY, DDMMYY

The following is a quoted string to allow spaces and commas etc within the format

DateFormats="D+ SHORTMONTH YYYY", "Date: D+ LONGMONTH, YYYY"

The following is for directory style dates

DateFormats=D+/M+/YY, MM/DD/YYYY

etc.

## Appendix D: Error messages

---

**(over socket): Error <error\_number>**

An error occurred while indexing into the DRE over the socket.

**(over socket): Error <error\_number>**

An error occurred while deleting files from the DRE.

**: Error <error\_number>**

An error occurred while indexing into DRE (on the same machine).

**<spider\_name> - indexing disabled (Couldn't find index host/port info), ignoring index command**

There is no valid DRE/Port combination.

**<spider\_name> is not a valid name for a spider.**

Spider names must not contain any of the following characters:

/ \ ; { } \* ? \ " !

**Connect Error : <error\_description>**

The spider encountered a connect error on a socket.

**Content length discrepancy for <URL>:**

The length of content returned by the server does not relate to the actual length returned.

**Couldn't connect.**

The spider could not connect to the server.

**Couldn't connect. Check your proxy settings.**

The spider could not connect to the site, probably due to the specified proxy settings.

**Couldn't determine file type of <URL>**

HTTPFetch could not find a valid file extension for the URL.

**Couldn't determine host. Check your proxy settings**

Either the host in the proxy settings is incorrect or the URL the spider is trying to retrieve is invalid.

**Couldn't get socket descriptor.**

The socket the spider is trying to use is invalid.

**Couldn't perform batch process at <number> pages.\n (Importing failed)**

The importing command failed.

**Couldn't read in configuration file <configuration\_file\_name>. Check permissions.**

The configuration file could not be read in.

**Couldn't recv.**

The spider could not receive from the server.

**Couldn't send.**

The spider could not send any data to the server.

**Couldn't setup connect**

The spider could not set the socket up from the parameters passed to it.

**Couldn't write page to disk**

The downloaded file could not be written to disk.

**Currently no files to import**

The fetch made a call to import zero files.

**ERROR ILLEGAL VALUE : SPIDERREPEATINTERVAL = <number>**

The repeat seconds were set to less than zero.

**ERROR: Different number of months for long and short formats**

The number of long months specified differs from the number of short months specified.

**Error: invalid spider directory**

The directory specified to place temporary files in does not exist.

**Error: License Invalid.**

The License is invalid.

**ERROR: No URL found for spider <spider\_name>.**

The spider has no URL configured for it.

**Exceeded maximum page duration on <URL>**

The page took too long to download.

**Exceeded page timeout on <URL>**

The page took too long to download.

**Failed to set the index to file output directory.**

The index directory for INDEXTOFILE could not be set.

**Fetch: Configuration file <config\_file\_name> not found**

HTTPFetch could not find the configuration file.

**Import: Couldn't create arguments.**

The import parameters could not be created.

**Import: Couldn't create import module.**

The import module could not be created.

**Import: Failure.**

The import module returned an error.

**Index: Couldn't get required parameters.**

The import/index directories are configured incorrectly.

**Index: Index file is zero length.**

The idx file is zero length.

**Index: No index file present to index.**

HTTPFetch is trying to index a nonexistent idx file.

**Indexing set to IndexToFile mode but no output dir has been specified. Please set the IndexDirectory value in the cfg file**

No directory has been specified for **IndexToFile**.

**Invalid configuration file name**

The configuration file has no name.

**MinPageSize larger than MaxPageSize, MinPageSize set to 0**

The minimum page size was set higher than the maximum page size.

**Recv Error on <URL>**

The content could not be read from the socket due to a server error.

**Recv Error on <URL> (Probably TimeOut/Socket Closed by Server).**

The spider encountered a receive error on a socket, probably due to the server.

**Select Error : <error\_description>**

The spider encountered a select error on a socket.

**Send Error : <error\_description>**

The spider encountered a send error on a socket.

**Send Error on <URL> :**

The spider could not send to the server.

**Spider: No active spiders configured.**

HTTPFetch could not create any spiders.

**The number of process's and extensions do not match, not applying**

The number of processes specified to apply before import do not match the number of extensions configured for them.

**Unhandled : <return\_code>**

The web server returned a return code that HTTPFetch could not understand.

**URL file <URL> does not exist.**

The specified URL file does not exist.

**Warning. Invalid License. Zero Users**

The license has zero users.

## Appendix D: Error messages

### **Web server <web\_server\_name> returned HTTP header containing invalid or missing modified date, using date now**

The web server either did not return a date or the returned date was invalid.

### **Web server <web\_server\_name> returned non-millennium compliant date format - Corrected**

The web server returned a non-millennium compliant date format, which was corrected.

### **Zero data length for URL <URL>**

The server did not return any data for the URL.



## Glossary

---

### **ACI (Autonomy Content Infrastructure)**

The Autonomy Content Infrastructure is a technology layer that automates operations on unstructured information for cross enterprise applications, thus enabling an automated and compatible business-to-business, peer-to-peer infrastructure.

The ACI allows enterprise applications to understand and process content that exists in unstructured formats, such as e-mail, Web pages, office documents, and Lotus Notes.

### **AXE (Autonomy XML Engine)**

The Autonomy XML Engine enables you to enhance the DRE's functionality by allowing you to input, output and process XML. It provides an infrastructure for complete automatic interoperability between applications using different tagging schemes, based on a conceptual understanding of XML documents, rather than on the tags themselves, and combines this with all other Autonomy functions.

### **Connector**

A Connector is an Autonomy fetching solution (for example HTTPFetch, Oracle Fetch, AutoIndexer and so on) that allows you to retrieve information from any type of local or remote repository (for example, a database or a web site). It imports the fetched documents into IDX or XML file format and indexes them into a DRE from where you can retrieve them (for example by sending queries to the DRE).

### **Database**

An Autonomy database is a data pool that is contained within the **DRE**. You can retrieve information that has been indexed into the **DRE** from the database by sending a query to the **DRE**.

### **DRE (Dynamic Reasoning Engine)**

The Dynamic Reasoning Engine is a scalable, multithreaded process which is based on advanced pattern-matching technology that exploits high-performance probabilistic modeling techniques. The DRE contains databases into which you can index information using a Connector, the DRE Administration tool or an indexing command. You can then access this information through a front end or by sending query commands to the DRE.

## **Fetching**

The process of downloading documents from the location they are stored in (for example a local folder, a website, a database, a Lotus Domino server and so on), importing them to IDX format and indexing them into a DRE.

## **Importing**

After a document has been downloaded from the location it is stored in, it is imported to an IDX file format. This process is called "importing".

## **IDX**

Apart from XML files, only files that are in IDX format can be indexed into a DRE. HTTPFetch imports files into this format. You can also manually index files.

## **Indexing**

The content (or links to the original documents) of IDX or XML documents is stored in a DRE. This process is called "indexing".

## **Query**

You can submit a natural language query to the DRE which analyzes the concept of the query and returns documents that are conceptually similar to the query's concept. You can also submit Boolean, bracketed Boolean and keyword searches to the DRE.

# Index

---

## A

ACI, 161  
 Adding a new site, 13  
 Administration Utility, 13, 59  
     Fetch Configuration Utility, 63  
 AfterDate, 124  
 AfterDate (Configuration setting), 36  
 Autonomy DiSH, 147  
 AXE, 161

## B

BatchProcess (Configuration setting), 41  
 BatchSize (Configuration setting), 41  
 BatchStartDeleteOld (Configuration setting), 41  
 BeforeDate, 124  
 BeforeDate (Configuration setting), 35

## C

CantHave, 120  
 CantHaveCheck (Configuration setting), 32  
 CantHaveCSVs (Configuration setting), 33  
 ClientCertificate (Configuration setting), 53  
 Configuration  
     Example, 127  
     Typical, 56  
 Configuration file, 24  
     [<MySpider>] section, 56  
     [Default] section, 26  
     [License] section, 24  
     [Service] section, 25  
     [Spider] section, 55  
     Example, 57  
 Configuration settings  
     <n>=<MyJobName>, 55  
     AfterDate, 36  
     BatchProcess, 41  
     BatchSize, 41  
     BatchStartDeleteOld, 41  
     BeforeDate, 35  
     CantHaveCheck, 32  
     CantHaveCSVs, 33  
     ClientCertificate, 53  
     CookieNamen, 32

CookieValuen, 32  
 Database, 26  
 DateCheck, 36  
 DateFormats, 38  
 DateLongMonthCSVs, 37  
 DateMonthCSVs, 38  
 DatePostfixCSVs, 39  
 DefaultDate, 39  
 DeleteByField, 41  
 DeleteStringCheck, 40  
 DeleteStringCSVs, 41  
 Depth, 30  
 Directory, 54  
 DREHost, 26  
 Extensions, 43  
 FollowRedirect, 31  
 FollowRobotMeta, 42  
 FollowRobotProtocol, 41  
 Holder, 24  
 HTMLImportStartDefFlags, 43  
 HTTPVersion, 43  
 IndexMode, 42  
 IndexOverSocket, 43  
 IndexPort, 26  
 Key, 24  
 LinkAllowCSVs, 45  
 LinkCheck, 44  
 LinkDisallowCSVs, 45  
 LogFile, 26  
 LogFileMaxSize, 26  
 LoginFieldNamen, 27  
 LoginFieldValuen, 27  
 LoginMethod, 27  
 LoginPassField, 28  
 LoginPassValue, 28  
 LoginURL, 28  
 LoginUserField, 28  
 LoginUserValue, 29  
 MaxGlobalKBPS, 29  
 MaxKBPS, 29  
 MaxLinksPerPage, 31  
 MaxPages, 31  
 MaxPageSize, 31  
 MaxRequests, 52  
 MaxRetries, 31  
 MinPageSize, 31  
 MustHaveCheck, 34  
 MustHaveCSVs, 35

- NavDirAllowCSVs, 46
- NavDirCheck, 45
- NavDirDisallowCSVs, 46
- NavLinkDefEndCSVs, 47
- NavLinkDefPostfixCSVs, 48
- NavLinkDefPrefixCSVs, 48
- NavLinkDefStartCSVs, 47
- NavLinksToFollow, 48
- NavSiteAllowCSVs, 49
- NavSiteCheck, 49
- NavSiteDisallowCSVs, 50
- NSockets, 29
- Number, 55
- PageDelay, 42
- PageDuration, 51
- PageTimeOut, 31
- ProxyHost, 51
- ProxyPassword, 51
- ProxyPort, 51
- ProxyUsername, 51
- SecurityType, 53
- ServiceControlClients, 25
- ServicePort, 25
- ServiceStatusClients, 25
- SiteDuration, 31
- SiteStructureCompactCopy, 43
- SpiderAccept, 51
- SpiderAs, 36
- SpiderCycles, 29
- SpiderPath, 53
- SpiderRedirectReplaceCSVs, 54
- SpiderRedirectReplaceWithCSVs, 54
- SpiderRepeatInterval, 51
- SpiderRepeatSecs, 30
- SpiderShowReferer, 52
- SpiderStartDeleteOld, 52
- SpiderStartTime, 29
- StayOnSite, 31
- StoreSiteStructure, 43
- TotalSize, 53
- TreatAsSlash, 53
- URLCaseSensitive, 52
- URLFile, 52
- URLFileDelete, 52
- URLn, 53
- Configuring the NTLM Proxy module, 152
- Configuring HTTPFetch, 23
  - Applying modifications, 23
  - Entering Boolean values, 23
  - Entering string values, 23

- Connector, 161
- CookieNamen (Configuration setting), 32
- CookieValuen (Configuration setting), 32

## D

- [Default] section (Configuration file), 26
- Database, 42, 161
- Database (Configuration setting), 26
- DateCheck (Configuration setting), 36
- DateFormats (Configuration setting), 38
- DateLongMonthCSVs (Configuration setting), 37
- DateMonthCSVs (Configuration setting), 38
- DatePostfixCSVs (Configuration setting), 39
- Dates, 123, 144, 146
  - Format of, 153
- DefaultDate (Configuration setting), 39
- DeleteByField (Configuration setting), 41
- DeleteStringCheck (Configuration setting), 40
- DeleteStringCSVs (Configuration setting), 41
- Depth (Configuration setting), 30
- Directory (Configuration setting), 54
- Directory structure
  - Unix, 12
  - Windows, 8
- Directory structures, 129
- Downloading, 118
- DRE, 1, 42, 76, 144, 146, 161, 162
- DREHost (Configuration setting), 26
- Dynamic web pages, 142

## E

- Error messages, 139, 155
- Example
  - Configuration, 127
  - Configuration file, 57
- Excluding pages, 119, 122
- Extensions (Configuration setting), 43

## F

- FAQ, 139
- Fetching, 162
- Fixed fields, 125
- FollowRedirect, 116
- FollowRedirect (Configuration setting), 31
- FollowRobotMeta (Configuration setting), 42
- FollowRobotProtocol (Configuration setting), 41

**H**

Holder (Configuration setting), 24  
 HTML, 134, 143  
 HTMLImportStartDefFlags (Configuration setting), 43  
 HTTPFetch, 1  
   Administration Utility, 59  
   Configuring, 23  
   Directory structure  
     Unix, 12  
     Windows, 8  
   FAQ, 139  
   Installing, 3  
   Troubleshooting, 135  
   Tutorial, 113  
 HTTPVersion (Configuration setting), 43

**I**

IDX, 129, 161, 162  
 Import Parameters, 72  
 Importing, 162  
 ImportStripLinks, 125  
 Indexing, 162  
 IndexMode (Configuration setting), 42  
 IndexOverSocket (Configuration setting), 43  
 IndexPort (Configuration setting), 26  
 Installation, 5, 10  
 Installing  
   The NTLM Proxy module, 149  
 Installing HTTPFetch, 3  
   Under Unix, 10  
   Under Windows, 5  
   With Portal-in-a-box, 3  
 Introduction, 1

**K**

Key (Configuration setting), 24

**L**

[License] section (Configuration file), 24  
 Languages, 62  
 Limits, 121  
   setting, 67  
   Setting, 70  
 LinkAllowCSVs (Configuration setting), 45  
 LinkCheck (Configuration setting), 44  
 LinkDisallowCSVs (Configuration setting), 45  
 Log files, 131

LogFile (Configuration setting), 26  
 LogFileMaxSize (Configuration setting), 26  
 LoginFieldName (Configuration setting), 27  
 LoginFieldValuen (Configuration setting), 27  
 LoginMethod (Configuration setting), 27  
 LoginPassField (Configuration setting), 28  
 LoginPassValue (Configuration setting), 28  
 LoginURL (Configuration setting), 28  
 LoginUserField (Configuration setting), 28  
 LoginUserValue (Configuration setting), 29

**M**

[<MySpider>] section (Configuration file), 56  
 MaxGlobalKBPS (Configuration setting), 29  
 MaxKBPS (Configuration setting), 29  
 MaxLinksPerPage (Configuration setting), 31  
 MaxPages, 120  
 MaxPages (Configuration setting), 31  
 MaxPageSize (Configuration setting), 31  
 MaxRequests (Configuration setting), 52  
 MaxRetries (Configuration setting), 31  
 Meta Tag, 126  
 MinPageSize (Configuration setting), 31  
 MustHave, 120  
 MustHaveCheck (Configuration setting), 34  
 MustHaveCSVs (Configuration setting), 35

**N**

<n>=<MyJobName> (Configuration setting), 55  
 NavDirAllowCSVs (Configuration setting), 46  
 NavDirCheck (Configuration setting), 45  
 NavDirDisallowCSVs (Configuration setting), 46  
 NavLinkDefEndCSVs (Configuration setting), 47  
 NavLinkDefPostfixCSVs (Configuration setting), 48  
 NavLinkDefPrefixCSVs (Configuration setting), 48  
 NavLinkDefStartCSVs (Configuration setting), 47  
 NavLinksToFollow (Configuration setting), 48  
 NavSiteAllowCSVs (Configuration setting), 49  
 NavSiteCheck (Configuration setting), 49  
 NavSiteDisallowCSVs (Configuration setting), 50  
 NSockets (Configuration setting), 29  
 NTLM, 22, 106  
 NTLM authentication, 150

## Index

NTLM Proxy module, 149  
  Configuring, 152  
  Finding out whether a site uses NTLM authentication, 150  
  Installing, 149  
  Pointing the HTTPFetch to the NTLM Proxy module, 151  
Number (Configuration setting), 55

## P

PageDelay (Configuration setting), 42  
PageDuration (Configuration setting), 51  
PageTimeout (Configuration setting), 31  
Pointing the HTTPFetch to the NTLM Proxy module, 151  
Portal-in-a-Box, 3  
ProxyHost (Configuration setting), 51  
ProxyPassword (Configuration setting), 51  
ProxyPort (Configuration setting), 51  
ProxyUsername (Configuration setting), 51

## Q

Query, 161, 162

## R

Redirection, 65  
Robots protocol, 65, 114

## S

[Service] section (Configuration file), 25  
[Spider] section (Configuration file), 55  
Scheduling, 117  
secure sites, 21  
SecurityType (Configuration setting), 53  
Service  
  Control Clients, 149  
  Port, 149  
ServiceControlClients (Configuration setting), 25  
ServicePort (Configuration setting), 25  
ServiceStatusClients (Configuration setting), 25  
SiteDuration, 116  
SiteDuration (Configuration setting), 31  
SiteStructureCompactCopy (Configuration setting), 43

Spider  
  Editing, 16  
  Setting up, 60  
SpiderAccept (Configuration setting), 51  
SpiderAs, 115  
SpiderAs (Configuration setting), 36  
SpiderCycles (Configuration setting), 29  
Spidering, 1, 113  
SpiderPath (Configuration setting), 53  
SpiderRedirectReplaceCSVs (Configuration setting), 54  
SpiderRedirectReplaceWithCSVs (Configuration setting), 54  
SpiderRepeatInterval (Configuration setting), 51  
SpiderRepeatSecs (Configuration setting), 30  
SpiderShowReferer (Configuration setting), 52  
SpiderStartDeleteOld (Configuration setting), 52  
SpiderStartTime (Configuration setting), 29  
StayOnSite (Configuration setting), 31  
StoreSiteStructure (Configuration setting), 43  
System architecture, 2  
System requirements, 3

## T

Temporary files, 129  
Testing configuration, 16  
TotalSize, 121  
TotalSize (Configuration setting), 53  
TreatAsSlash (Configuration setting), 53  
Troubleshooting, 135  
Tutorial, 113

## U

URLCaseSensitive (Configuration setting), 52  
URLFile (Configuration setting), 52  
URLFileDelete (Configuration setting), 52  
URLn (Configuration setting), 53

## V

Variable fields, 125

## X

XML, 89