# Endeca Content Acquisition System

## Web Crawler Guide

Version 3.0.2 • March 2012

ORACLE®

ENDECA

# Contents

# Copyright and disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

# Preface

The Oracle Endeca Commerce solution enables your company to deliver a personalized, consistent customer buying experience across all channels — online, in-store, mobile, or social. Whenever and wherever customers engage with your business, the Oracle Endeca Commerce solution delivers, analyzes, and targets just the right content to just the right customer to encourage clicks and drive business results.

Oracle Endeca Commerce is the most effective way for your customers to dynamically explore your storefront and find relevant and desired items quickly. An industry-leading faceted search and Guided Navigation solution, Oracle Endeca Commerce enables businesses to help guide and influence customers in each step of their search experience. At the core of Oracle Endeca Commerce is the MDEX Engine™, a hybrid search-analytical database specifically designed for high-performance exploration and discovery. The Endeca Content Acquisition System provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. Endeca Assembler dynamically assembles content from any resource and seamlessly combines it with results from the MDEX Engine.

Oracle Endeca Experience Manager is a single, flexible solution that enables you to create, deliver, and manage content-rich, cross-channel customer experiences. It also enables non-technical business users to deliver targeted, user-centric online experiences in a scalable way — creating always-relevant customer interactions that increase conversion rates and accelerate cross-channel sales. Non-technical users can control how, where, when, and what type of content is presented in response to any search, category selection, or facet refinement.

These components — along with additional modules for SEO, Social, and Mobile channel support — make up the core of Oracle Endeca Experience Manager, a customer experience management platform focused on delivering the most relevant, targeted, and optimized experience for every customer, at every step, across all customer touch points.

## About this guide

This guide describes the major tasks involved in configuring the Endeca Web Crawler and running it to gather source data from Web sites.

It assumes that you are familiar with the concepts of the Endeca Content Acquisition System and the Endeca Information Transformation Layer.

## Who should use this guide

This guide is intended for application developers who are building applications using the Endeca Web Crawler and are responsible for running Web crawls and providing the data into an Endeca pipeline to transform them into Endeca records.

# Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: ¬

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

# Contacting Oracle Support

Oracle Support provides registered users with important information regarding Oracle Endeca software, implementation questions, product and solution help, as well as overall news and updates.

You can contact Oracle Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Chapter 1

# Introduction

This section provides introductory information on the Endeca Web Crawler.

## Web Crawler Overview

The Web Crawler is installed by default as part of the CAS installation. The Endeca Web Crawler gathers source data by crawling HTTP and HTTPS Web sites and writes the data in a format that is ready for Forge processing (XML or binary).

Besides retrieving and converting the source documents, the Web Crawler tags the resulting Endeca records with metadata properties that are derived from the source documents.

After the Web Crawler writes the Endeca records, you can configure an Endeca record adapter (in Developer Studio) to read the records into your Endeca pipeline, where Forge processes the records, and you can add or modify the record properties. These property values can then be mapped to Endeca dimensions or properties by the property mapper in the pipeline. For details, see "Creating a Pipeline to read Endeca records" in the *Endeca CAS Developer's Guide*.

You can then build an Endeca application to access the records and allow your application users to search and navigate the document contents contained in the records.

The Endeca Web Crawler is intended for large-scale crawling and is designed with a highly modular architecture that allows developers to create their own plugins. The Endeca Web Crawler supports these types of crawls:

- **full** crawls, in which all pages (URLs) in the seed are crawled.
- **resumable** crawls (also called restartable crawls), in which the crawl uses the same seed as a previous crawl, but uses a different crawl depth or configuration.

Note that the current version of the Endeca Web Crawler does not support incremental crawls nor crawling FTP sites.

### SSL Support

You can configure the Endeca Web Crawler to read and write from an SSL-enabled Record Store instance. For details, see the "SSL Configuration" chapter of the *Endeca CAS Developer's Guide*.

# Running the Endeca sample Web crawl

You can examine the configuration and operation of the Web Crawler by running a sample Web crawl located in the `CAS\workspace\conf\web-crawler\polite-crawl` directory.

The sample configuration crawls the Endeca Web site (`http://www.endeca.com`) with a preconfigured seed file (`endeca.lst`) in the `conf\web-crawler\default` directory.

The Endeca sample crawl is configured to output the records as uncompressed XML. The XML format allows you to easily read the output file (with a text editor or the `more` command) to confirm that the crawl collected records. The `site.xml` file also specifies `polite-crawl-workspace` as the name of the workspace directory.

**To run the Endeca sample crawl:**

1. Open a command prompt.
2. Navigate to the CAS root directory.

   For example, in a default installation on Windows, this is `C:\Endeca\CAS\`*`version`* .
3. Run the `web-crawler.bat` (for Windows) or `web-crawler.sh` (for UNIX) script with the following flags. Be sure to specify 0 (zero) to the `-d` flag to crawl only the root of the site, as shown in this example on a Windows machine:

   ```
   .\bin\web-crawler -c ..\workspace\conf\web-crawler\polite-crawl
   -d 0 -s http://www.endeca.com
   ```

   If the crawl begins successfully, you see the `INFO` progress messages.

When finished, the Web Crawler displays: `Crawl complete`. The output file named `polite-crawl.xml` is in the `CAS\`*`version`*`\polite-crawl-workspace\output` directory.

Chapter 2

# Configuration

This section provides configuration information for the Endeca Web Crawler.

## Configuration files

The Endeca Web Crawler uses the following set of configuration files:

| Configuration Filename | Purpose |
|---|---|
| `default.xml` | The global configuration file, which should contain properties for all of your crawls with reasonable default values. Specific settings in this file can be overridden by the `site.xml` file. Do not remove or rename this file, because its name and location are hard-coded in the Web Crawler software. |
| `site.xml` | A per-crawl property overrides file. The settings in this file override those in the `default.xml` file. Therefore, this file is meant to be used to adjust per-crawl settings. |
| `crawl-urlfilter.txt` | Contains a list of include and exclude regular expressions for URLs. These expressions determine which URLs the crawler is allowed to visit. Note that the filters can also be applied to seeds if the `urlfilter.filter-seeds` configuration property is set to `true`. |
| `regex-normalize.xml` | Contains a list of URL normalizations, which allow you to specify substitutions to be done on URLs. Each normalization is expressed as a regular expression and a replacement expression. Note that the seeds can also be normalized if the `urlnormalizer.normalize-seeds` configuration property is set to `true`. |
| `mime-types.xml` | Contains a list of MIME types known to the system. It is used to look up the MIME type for a specific file extension. |
| `parse-plugins.xml` | Maps MIME types to parsers (for example, "text/html" to the HTML parser). |
| `form-credentials.xml` | The credentials file for form-based authentication. |
| `log4j.properties` | The log4j configuration file, which is used to specify logging on certain components. |

### Location of the configuration files

After you install the CAS, the configuration files are in the following locations:

- The `workspace/conf/web-crawler/default` directory contains all of the above files, except for the `site.xml` file. This directory is the global configuration directory, and you should not change its name nor remove the `default.xml` file. Note that the settings of most of its files can be overridden by the versions in the crawl-specific configuration directories.
- The `workspace/conf/web-crawler/polite-crawl` directory contains only the `site.xml` and `crawl-urlfilter.txt` files.
- The `workspace/conf/web-crawler/non-polite-crawl` directory also contains only the `site.xml` and `crawl-urlfilter.txt` files. This `site.xml` contains more aggressive settings, such as such as no fetcher delay (versus a 1-second delay in the polite version) and a maximum of 52 threads (versus 1 in the polite version).

You can use a text editor to edit the files.

# The default.xml file

The `default.xml` file is the main configuration file for the Endeca Web Crawler.

The `default.xml` configuration file contains properties for all of your crawls. These properties should have values that can be used for most crawl scenarios. If necessary, you can override these default values with those in the `site.xml` file.

The `default.xml` file provides configuration values for these sets of properties:

- HTTP properties
- Authentication properties
- Proxy properties
- Fetcher properties
- URL normalization properties
- MIME type properties
- Plugin properties
- Parser properties
- Parser filter properties
- URL filter properties
- Crawl scoping properties
- Document Conversion properties
- Output file properties

Each set of properties is covered in its own topic page.

**Note:** Do not change the name or location of the `default.xml` configuration file because the Web Crawler is hard-coded to look for that name and path. If you rename the file, the Web Crawler throws an exception at start-up and exit.

## HTTP Properties

You can set the HTTP properties in the `default.xml` file.

The `default.xml` configuration file allows you to set the HTTP transport properties for the Web Crawler.

| Property Name | Property Value |
|---|---|
| `http.agent.name` | String that contains the name of the user agent originating the request (default is `endeca webcrawler`). This value is used for the HTTP User-Agent request header. Required. |
| `http.robots.ignore` | Boolean value (default is `false`). Determines whether the crawler ignores the `robots.txt`. |
| `http.robots.agents` | Comma-delimited list of agent strings, in decreasing order of precedence (default is `endeca webcrawler,*`). The agent strings are checked against the User-Agent field in the `robots.txt` file. It is recommended that you put the value of `http.agent.name` as the first agent name and keep the asterisk (*) at the end of the list. |
| `http.robots.403.allow` | Boolean value (default is `false`). Some servers return HTTP status 403 (Forbidden) if `robots.txt` does not exist. Setting this value to `false` means that such sites are treated as forbidden, while setting it to `true` means that the site can be crawled. |
| `http.agent.description` | String value (default is empty). Provides descriptive text about the crawler. The text is used in the User-Agent header, appearing in parenthesis after the agent name. |
| `http.agent.url` | String value (default is empty). Specifies the URL that appears in the User-Agent header, in parenthesis after the agent name. Custom dictates that the URL be a page explaining the purpose and behavior of this crawler. |
| `http.agent.email` | String value (default is empty). Specifies the email address that appears in the HTTP From request header and User-Agent header. A good practice is to mangle this address (e.g., "info at example dot com") to avoid spamming. |
| `http.agent.version` | String value (default is `WebCrawler`). Specifies the version of the crawl. The version is used in the User-Agent header. |
| `http.timeout` | Integer value (default is `10000`). Specifies the default network timeout in milliseconds. |
| `http.content.limit` | Integer value (default is `1048576`). Sets the length limit in bytes for downloaded content. If the value is a positive integer greater than 0, content longer than the setting will not be downloaded (the page will be skipped). If set to a negative integer, no limit is set on the content length. Endeca does not recommend setting this value to `0` because that value limits the crawl to producing 0-byte content. |
| `http.redirect.max` | Integer value (default is `5`). Sets the maximum number of redirects the fetcher will follow when trying to fetch a page. If set to negative or 0, the fetcher will not immediately follow redirected URLs, but instead will record them for later fetching. |

| Property Name | Property Value |
|---|---|
| `http.useHttp11` | Boolean value (default is `false`). If `true`, use HTTP 1.1; if `false`, use HTTP 1.0. |
| `http.cookies` | String value (default is empty). Specifies the cookies to be used by the HTTPClient. |

## About setting the HTTPClient cookies

The `http.cookies` property sets the cookies used by the HTTPClient.

The cookies must be in this format:

```
DOMAIN1~~~NAME1~~~VALUE1~~~PATH1~~~MAXAGE1~~~SECURE1|||DOMAIN2~~~...
```

where:

- `DOMAIN` is the domain the cookie can be sent to.
- `NAME` is the cookie name.
- `VALUE` is the cookie value.
- `PATH` is the path prefix for which the cookie can be sent.
- `MAXAGE` is the number of seconds for which the cookie is valid (expected to be a non-negative number, -1 signifies that the cookie should never expire).
- `SECURE` is either `true` (the cookie can only be sent over secure connections, that is, HTTPS servers) or `false` (the cookie is considered safe to be sent in the clear over unsecured channels).

Note that the triple-tilde delimiter (~~~) must be used to separate the values.

A sample cookie specification is:

```
172.30.112.218~~~MYCOOKIE~~~ABRACADABRA=MAGIC~~~/junglegym/mycookie.jsp~~~-
1~~~false
```

Note that the example cookie never expires and can be sent over unsecured channels.

## About obeying the robots.txt file

You can set the Web Crawler to either ignore or obey the `robots.txt` exclusion standard, as well as any META ROBOTS tags in HTML pages.

**Note:** By default, the `http.robots.ignore property` is set to `false` in `default.xml`. However, `site.xml` in the `conf/web-crawler/non-polite-crawl` directory contains an override for the `http.robots.ignore` property, which is set to `true` in that file.

For example, if the property is set to `false` and an HTML page has these META tags:

```
<html>
<head>
<title>Sample Page</title>
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
</head>
```

then the presence of the NOINDEX tag causes the crawler to not index the content of the page (i.e., no text or title is extracted), while the NOFOLLOW tag prevents outlinks from being extracted from the page. In addition, a message is logged for each META tag that is obeyed:

```
The HTML meta tags for robots contains "noindex", no text and title are
extracted for: URL
```

```
The HTML meta tags for robots contains "nofollow", no outlinks are extracted
 for: URL
```

If the property is set to `true`, then the `robots.txt` file is ignored, as well as any META ROBOTS tags in HTML pages (for example, outlinks are extracted even if the META ROBOTS tag is set to NOFOLLOW).

## Setting the download content limit

If your crawls are downloading files with a lot of content (for example, large PDF or SWF files), you may see WARN messages about pages being skipped because the content limit was exceeded. To solve this problem, you should increase the download content limit to a setting that allows all content to be downloaded.

Any content longer than the size limit is not downloaded (i.e., the page is skipped).

To set the download content limit:

1. In a text editor, open `default.xml`.
2. Set the value of the `http.content.limit` property as the length limit, in bytes, for download content.

   **Note:** Note that if the content limit is set to a negative number or 0, no limit is imposed on the content. However, this setting is not recommended because the Web Crawler may encounter very large files that slow down the crawl.

3. Save and close the file.

### Example of setting the download content limit

In this example, the size of the content is larger than the setting of the `http.content.limit` property:

```
WARN com.endeca.itl.web.UrlProcessor
Content limit exceeded for http://xyz.com/pdf/B2B_info.pdf. Page is skipped.
```

# Authentication properties

You can set the authentication properties in the `default.xml` file.

The HTTPClient supports four different types of HTTP authentication schemes :

- Basic
- Digest
- NTLM
- Form

These schemes can be used to authenticate with HTTP servers or proxies. The table below lists the properties that correspond to each authentication scheme.

| Property Name | Property Value |
|---|---|
| `http.auth.basic` | String value (default is empty). Specifies the credentials to be used by the HTTPClient for Basic authentication. If the value is empty, Basic authentication is not done for the crawl. |
| `http.auth.digest` | String value (default is empty). Specifies the credentials to be used by the HTTPClient for Digest authentication. If the value is empty, Digest authentication is not done for the crawl. |
| `http.auth.ntlm` | String value (default is empty). Specifies the credentials to be used by the HTTPClient for NTLM authentication. If the value is empty, NTLM authentication is not done for the crawl. |
| `http.auth.form.credentials.file` | File name (default is `form-credentials.xml`). Specifies the file in the configuration directory that provides the credentials for Form-based authentication. If the value is empty, Form authentication is not done for the crawl. |

**Related Links**

*The form-credentials.xml file* on page 38

The `form-credentials.xml` file provides the credentials for sites that use form-based authentication.

## About configuring Basic authentication

If a Web server uses HTTP Basic authentication to restrict access to Web sites, you can specify authentication credentials that enable the Web Crawler to access password-protected pages. The `http.auth.basic` property sets the credentials to be used by the HTTPClient for Basic authentication.

The credentials must be specified in this format :

```
USERNAME1~~~PASSWORD1~~~HOST1~~~PORT1~~~REALM1|||USERNAME2~~~...
```

where:

- `USERNAME` is the user ID to be sent to the host server.
- `PASSWORD` is the password for the user ID.
- `HOST` is the host to which the credentials apply (i.e., the host to be crawled). The value can be a specific host name or `ANY_HOST` (which represents any host).
- `PORT` is either a specific host port or `ANY_PORT`.
- `REALM` is either a specific realm name on the host or `ANY_REALM`.

Note that the triple-tilde delimiter (~~~) must be used to separate the values.

A sample credential specification is:

```
jjones~~~hello123~~~myhost~~~ANY_PORT~~~ANY_REALM
```

## About configuring Digest authentication

If a Web server uses HTTP Digest authentication to restrict access to Web sites, you can use the `http.auth.digest` property to set the credentials used by the HTTPClient for Digest authentication.

The credentials must be specified in this format:

```
USERNAME1~~~PASSWORD1~~~HOST1~~~PORT1~~~REALM1|||USERNAME2~~~...
```

where the meanings of the arguments are the same as for Basic authentication.

### About configuring NTLM authentication

If a Web server uses HTTP NTLM authentication to restrict access to Web sites, you can specify authentication credentials that enable the Web Crawler to access password-protected pages. The `http.auth.ntlm` property sets the credentials to be used by the HTTPClient for NTLM authentication.

**Note:** The Web Crawler only supports Version 1 of the NTLM authentication scheme.

The credentials must be specified in this format:

```
USERNAME1~~~PASSWORD1~~~HOST1~~~PORT1~~~REALM1~~~DOMAIN1|||USERNAME2~~~...
```

where:

- `USERNAME` is the user ID to be sent to the server.
- `PASSWORD` is the password for the user ID.
- `HOST` is a specific host name to which the credentials apply (i.e., the host to be crawled). Note that you cannot use the `ANY_HOST` specifier.
- `PORT` is either a specific host port or `ANY_PORT`.
- `REALM` is either a specific realm name on the host or `ANY_REALM`.
- `DOMAIN` is either a domain name or an IP address.

Note that the triple-tilde delimiter (~~~) must be used to separate the values.

### Configuring Form-based authentication

If you are crawling sites that implement form-based authentication, you supply the credentials in a `form-credentials.xml` file.

To configure form-based authentication:

1. In a text editor, open `default.xml`.
2. Use the `http.auth.form.credentials.file` property to specify the name of the `form-credentials.xml` file.

   **Note:** The `form-credentials.xml` file should be located in either `workspace/conf/web-crawler/default` or the directory that holds a per-crawl set of configuration files.

## Properties for authenticated proxy support

You can configure authenticated proxy support for the Web Crawler.

Many networks use authenticated proxy servers to secure and control Internet access. These proxy servers require a unique user ID and password for access.

| Property Name | Property Value |
|---|---|
| `http.proxy.host` | String value (default is empty). Specifies the hostname of the authenticated proxy server. If the value is empty, no proxy is used. |
| `http.proxy.port` | Number that specifies the port of the authenticated proxy server (default is empty). |
| `http.proxy.agent.host` | Name or IP address of the host on which the crawler would be running (default is empty). This value is used by the protocol-httpclient plugin. Use this property only if the proxy needs NTLM authentication. |
| `http.proxy.username` | String value (default is empty). Specifies the username of the proxy. The name will be used by the protocol-httpclient plugin, if the proxy server requests basic, digest, and/or NTLM authentication. For NTLM authentication, do not prefix the username with the domain (`susam` is correct whereas `DOMAIN\susam` is incorrect). |
| `http.proxy.password` | String value (default is empty). Specifies the password for the proxy user ID. |
| `http.proxy.realm` | String value (default is empty). Specifies the authentication realm for the proxy. Do not specify a value if a realm is not required or if authentication should take place for any realm. If the site is using NTLM authentication, note that NTLM does not use the notion of realms; therefore, you must specify the domain name of NTLM authentication as the value for this property. |

## Fetcher properties

The fetcher is the Web Crawler component that actually fetches pages from Web sites. You can set the fetcher properties in the `default.xml` file.

By using the properties listed in the table, you can configure the behavior of the fetcher.

| Property Name | Property Value |
|---|---|
| `fetcher.delay` | Value in seconds (default is `2.0`). Specifies the number of seconds a fetcher will delay between successive requests to the same server. If you have multiple threads per host, the delay is on a per-thread basis, not across all threads. |
| `fetcher.delay.max` | Value in seconds (default is `30`). Specifies the maximum amount of time to wait between page requests. |
| `fetcher.threads.total` | Integer (default is `100`). Specifies the number of threads the fetcher should use. This value also |

| Property Name | Property Value |
|---|---|
| | determines the maximum number of requests that are made at once (because each thread handles one connection). |
| `fetcher.threads.per-host` | Integer (default is `1`). Specifies the maximum number of threads that should be allowed to access a host at one time. |
| `fetcher.retry.max` | Integer (default is `3`). Specifies the maximum number of times that a page will be retried. The page is skipped if it cannot be fetched in this number of retries. |
| `fetcher.retry.delay` | Value in seconds (default is `5`). Specifies the delay between subsequent retries on the same page. If this value is less than the `fetcher.delay` value, then the value of `fetcher.delay` is used instead. |

## Use of the max delay and crawl-delay values

The fetcher compares the value of the `fetcher.delay.max` property to the value of the Crawl-Delay parameter in the `robots.txt` file.

The fetcher works as follows:

- If the `fetcher.delay.max` value is greater than the Crawl-Delay value, the fetcher will obey the amount of time specified by Crawl-Delay.
- If the `fetcher.delay.max` value is less than the Crawl-Delay value, the fetcher will not crawl the site. It will also generate this error message:

  ```
  The delay specified in robots.txt is greater than the max delay.
  Therefore the crawler will not fully crawl this site. All pending work
  from this host has been removed.
  ```

- If the `fetcher.delay.max` value is set to `-1`, the fetcher will wait the amount of time specified by the Crawl-Delay value.

Note that above behavior occurs only if the `http.robots.ignore` property is set to `false` (which is the default).

## Fetcher overrides in the site.xml files

This topic describes overrides for the fetcher property values in the `default.xml` file.

The `site.xml` file in the `workspace/conf/web-crawler/non-polite-crawl` directory contains overrides to the fetcher's default property values.

- The `fetcher.delay` value is set to `0.0`.
- The `fetcher.threads.total` value is set to `52`.
- The `fetcher.threads.per-host` value is set to `52`.

The `site.xml` file in the `workspace/conf/web-crawler/polite-crawl` directory overrides the `fetcher.delay` value, which it sets to `1.0`.

Otherwise, both files use the default values for the fetcher properties.

## URL normalization properties

You can set the URL normalization properties in the `default.xml` file.

URL normalization (also called URL canonicalization) is the process by which URLs are modified and standardized in a consistent manner. The purpose of URL normalization is to transform a URL into a normalized or canonical URL so it is possible to determine if two syntactically different URLs are equivalent.

The Web Crawler performs URL normalization in order to avoid crawling the same resource more than once. By using the properties listed in the table, you can configure how the Web Crawler normalizes URLs.

| Property Name | Property Value |
|---|---|
| `urlnormalizer.order` | Space-delimited list of URL normalization class names. Specifies the order in which the URL normalizers will be run. If any normalizer is not activated, it will be silently skipped. If other normalizers not on the list are activated, they will run in random order after the listed normalizers run. |
| `urlnormalizer.regex.file` | File name (default is `regex-normalize.xml`). Name of the configuration file used by the `Regex¬UrlNormalizer` class. Note that the file must be in the configuration directory. |
| `urlnormalizer.loop.count` | Integer value (default is `1`). Specifies how many times to loop through normalizers, to ensure that all transformations are performed. |
| `urlnormalizer.normalize-seeds` | Boolean value (default is `false`). Specifies whether to normalize the seeds. |

### Types of URL normalizers

The Endeca Web Crawler has three URL normalizers:

- `BasicURLNormalizer`
- `PassURLNormalizer`
- `RegexURLNormalizer`

The BasicURLNormalizer performs the following transformations:

- Removes leading and trailing white spaces in the URL.
- Lowercases the protocol (e.g., `HTTP` is changed to `http`).
- Lowercases the host name.
- Normalizes the port (e.g., `http://xyz.com:80/index.html` is changed to `http://xyz.com/index.html`).
- Normalizes null paths (e.g., `http://xyz.com` is changed to `http://xyz.com/index.html`).
- Removes references (e.g., `http://xyz.com/about.html#history` is changed to `http://xyz.com/about.html`).
- Removes unnecessary paths, in particular the `../` paths.

Note that these transformations are actually performed by the `regex-normalize.xml` file.

The `PassURLNormalizer` performs no transformations. It is included because it is sometimes useful if for a given scope at least one normalizer must be defined but no transformations are required.

The `RegexURLNormalizer` allows users to specify regex substituions on all or any URLs that are encountered. This is useful for transformations like stripping session IDs from URLs. This class uses the file specified in the `urlnormalizer.regex.file` property.

### Default order for the URL normalizers

The default classes for the `urlnormalizer.order` property are:

- `org.apache.nutch.net.urlnormalizer.basic.BasicURLNormalizer`
- `org.apache.nutch.net.urlnormalizer.regex.RegexURLNormalizer`

## Normalizing the seed list

You can apply normalization to the seed list with the `urlnormalizer.normalize-seeds` property.

By default, the seeds are read in as-is. In some cases, however, you may want to have URL normalization applied to the seeds (for example, if the seeds are extracted from a database instead of manually entered in the seed list by the user).

To normalize the seed list:

1. In a text editor, open the `default.xml` file.
2. Set the `urlnormalizer.normalize-seeds` property to `true`.
3. Save and close the file.

**Related Links**

*URL filter properties* on page 25

You can configure how the URL filter plugins are handled by the Web Crawler.

# MIME type properties

You can set the MIME type mapping properties in the `default.xml` file.

These properties provide a high-level configuration of how the Web Crawler performs the mapping of file extensions to MIME types. Note that by default, the list of MIME file extensions is kept in the `mime-types.xml` configuration file.

| Property Name | Property Value |
|---|---|
| `mime.types.file` | File name (default is `mime-types.xml`). Specifies the file in the configuration directory that contains information mapping filename extensions and magic sequences to MIME types. |
| `mime.type.magic` | Boolean value (default is `true`). Specifies whether the MIME content-type detector uses magic resolution to determine the MIME type. |
| `mime.types.trust-server.text-html` | Boolean value (default is `false`). Specifies whether the "text/html" MIME type returned by the server should be trusted over the URL extension. |

### Overriding the server text/html MIME type

If there is confusion as to the MIME type of a given URL, the Web Crawler by default trusts the URL extension over the server MIME type. The `mime.types.trust-server.text-html` property is intended for crawls that may experience "text/html" MIME type resolution problems.

Assume, for example, that one of the URLs to be crawled is similar to the following:

```
http://www.xyz.com/scripts/InfoPDF.asp?FileName=4368.pdf
```

In this case, the actual page is an ASP page, and therefore the server returns "text/html" as the MIME type for the page. However, the crawler sees that the URL has a ".pdf" extension, and therefore resolves it as a PDF file (i.e., it overrides the MIME type returned by the server). The crawler then invokes the Document Conversion module on the page, when in fact it should not.

In the above example, if the `mime.types.trust-server.text-html` property is set to `true`, the crawler trusts the server's "text/html" MIME type instead of the URL extension when resolving this contention. The Document Conversion module is therefore not invoked.

To override the server text/html MIME type:

1. In a text editor, open the `default.xml` file.
2. Set the `mime.types.trust-server.text-html` property to `true`.
3. Save the file.

## Plugin properties

You can set the plugin properties in the `default.xml` file.

The Web Crawler contains a number of plugins that perform the core work of the crawler tasks. By using the properties listed in the table, you can configure which plugins to activate and how to handle non-activated plugins that are needed by activated plugins.

| Property Name | Property Value |
|---|---|
| `plugin.folders` | Comma-delimited list of directory pathnames (default is `CAS/`*`version`*`/lib/web-crawler/plugins`). Specifies the directories where the plugins are located. Each element may be a relative or absolute path. If absolute, it is used as-is; If relative, it is searched for on the CLASSPATH. |
| `plugin.auto-activation` | Boolean value (default is `true`). Specifies if some plugins that are not activated by the `plugin.includes` and `plugin.excludes` properties must be automatically activated if they are needed by some activated plugins. |
| `plugin.includes` | Regular expression. Specifies which plugin IDs to include. Any plugin not matching this expression is excluded. |
| `plugin.excludes` | Regular expression (default is empty). Specifies which plugin IDs to exclude. |

## Default activated plugins

The default regular expression value for the `plugin.includes` property activates these plugins:

- lib-auth-http
- auth-http-form-basic
- protocol-httpclient
- protocol-file
- urlfilter-regex
- parse-text
- parse-html
- parse-js
- urlnormalizer-pass
- urlnormalizer-regex
- urlnormalizer-basic
- endeca-searchexport-converter-parser
- endeca-generator-html-basic
- output-endeca-record

## Specifying the plugins directory

The `plugin.folders` property lets you specify the location of the plugins directory.

If you retain the default `.../lib/web-crawler/plugins` location, you have to run the `web-crawler` startup script from the Web Crawler's root directory. If you specify an abolute path for the location, you can run the script from any other directory on the machine.

To specify the plugins directory:

1. In a text editor, open the `default.xml` file.
2. Modify the `plugin.folders` property as needed.
3. Save and close the file.

# Parser properties

You can set the parser properties in the `default.xml` file.

The Web Crawler contains two HTML scanners that parse HTML documents: NekoHTML and TagSoup. By using the properties listed in the table, you can configure which HTML parser to use, as well as other parsing behavior.

| Property Name | Property Value |
|---|---|
| `parse.plugin.file` | File name (default is `parse-plugins.xml`). Specifies the configuration file that defines the associations between content-types and parsers. |
| `parser.character.encoding.default` | ISO code or other encoding representation (default is `windows-1252`). Specifies the character encoding to use when no other information is available. |

| Property Name | Property Value |
|---|---|
| `parser.html.impl` | `neko` or `tagsoup` (default is `neko`). Specifies which HTML parser implementation to use: `neko` uses NekoHTML and `tagsoup` uses TagSoup. |
| `parser.html.form.use_action` | Boolean value (default is `false`). If `true`, the HTML parser will collect URLs from Form action attributes.<br>**Note:** This may lead to undesirable behavior, such as submitting empty forms during the next fetch cycle.<br>If `false`, form action attributes will be ignored. |

If the Web Crawler configuration includes the DOM for the Web page in the output Endeca records, the HTML parsers handle invalid XML characters as follows:

- The NekoHTML parser removes the invalid XML characters in the range 0x00-0x1F and 0x7F-0x9F from the DOM.
- The TagSoup parser strips nothing from the DOM, because TagSoup can efficiently handle invalid XML characters.

Note that the NekoHTML parser is the default HTML parser.

## Parser filter properties

You can set the parser filter properties in the `default.xml` file.

The Web Crawler contains a number of filter plugins that perform the core work of the crawler tasks. By using the properties listed in the table, you can configure how the plugins are handled by the Web Crawler.

| Property Name | Property Value |
|---|---|
| `parser.filters.order` | Space-delimited list of parser filter class names (default is empty). Specifies the order in which the parser filters are applied. |
| `document.prune.xpath` | String of XPath expressions (default is empty). Defines the XPath expressions to be used for the endeca-xpath-filter. |
| `document.prune.xpath.follow-outlinks` | Boolean value (default is `true`). Determines whether the crawler will follow outlinks from the pruned content. If set to `true` (the default), the outlinks are followed.<br>**Note:** To use this feature, you must include `endeca-xpath-filter` in the `plugin.includes` property. |

## Setting the order of parser filters

The `parser.filters.order` property allows you to specify the order in which the parser filters are applied.

To set the order of parser filters:

1. In a text editor, open the `default.xml` file.
2. Modify the `parser.filters.order` property as needed.

If the property value is empty, all available parser filters (as dictated by the `plugin-includes` and `plugin-excludes` properties) are loaded and applied in system-defined order.

If the property value is not empty, only the named filters are loaded and applied in the given order. For example, assume that the property has this value:

```
org.apache.nutch.parse.js.JSParseFilter com.endeca.itl.web.process.filter.DocumentPruneXPathFilter
```

In this case, the JSParseFilter is applied first and the DocumentPruneXPathFilter second.

## About defining the XPath filter expressions

The `document.prune.xpath` property defines the XPath expressions that will be used by the Endeca Document Prune XPath Filter (i.e., the endeca-xpath-filter plugin).

The XPath expressions are delimited using a triple-tilde delimiter (~~~) and are used to prune the document in this order. Note that all the element names must be defined in uppercase while the attribute names must be in lowercase.

**Note:** To use this property, you need to include `endeca-xpath-filter` in the `plugin.includes` property.

Example 1: Assume that the property has this XPath expression value:

```
//DIV~~~//A[@href]
```

This expression would prune all the DIV elements and links (i.e., the A anchor elements) in the document.

Example 2: Assume that many of the pages that you are crawling have the same header and footer. Because the text that is in the header and footer has no correlation to the subject matter of the page, you want to prune the header and footer text. The XPath expression for this operation would look similar to this example:

```
//DIV[@id="masthead"]~~~//DIV[@class="flash"]~~~//DIV[@id="header"]~~~
//DIV[@id="footer"]~~~//SCRIPT~~~//DIV[@id="breadcrumbs"]~~~//DIV[@id="clearBoth"]
```

**Note:** If the headers and footers are links, you can set the `document.prune.xpath.follow-outlinks` property to `false` to also prune all outlinks.

# URL filter properties

You can configure how the URL filter plugins are handled by the Web Crawler.

| Property Name | Property Value |
|---|---|
| `urlfilter.regex.file` | File name (default is `crawl-urlfilter.txt`). Specifies the file in the configuration directory containing regular expressions used by the urlfilter-regex (RegexURLFilter) plugin. |
| `urlfilter.order` | Space-delimited list of URL filter class names (default is empty). Specifies the order in which URL filters are applied. |
| `urlfilter.filter-seeds` | Boolean value (default is `false`). Specifies whether URL filtering should be applied to the seeds. |

### Interaction with crawl scope filtering

Keep in mind that the crawl scope filter (if configured) is applied before all other filters including the regular expressions in this file custom plugins. This means that once a URL has been filtered out by the crawl scope, it cannot be added by expressions in this file.

## Setting the order of URL filters

The `urlfilter.order` property allows you to specify the order in which URL filters are applied.

If the property value is empty, all available URL filters (as dictated by the `plugin-includes` and `plugin-excludes` properties) are loaded and applied in system-defined order. If the property value is not empty, only the named filters are loaded and applied in the given order.

To set the order of URL filters:

1. In a text editor, open `default.xml`.
2. Set the value of the `urlfilter.order` property as a space delimited list of URL filters in order of priority.
3. Save and close the file.

### Example of setting the order of URL filters

Assume that the `urlfilter.order` property has this value:

```
org.apache.nutch.urlfilter.regex.RegexURLFilter sample.project.urlfilter.sam¬
ple.SampleFilter
```

In this case, the RegexURLFilter is applied first and the SampleFilter second.

Because all filters are AND'ed, filter ordering does not have an impact on the end result. However, it may have a performance implication, depending on the relative expensiveness of the filters.

## Filtering the seed list

You can apply URL filtering to the seeds with the `urlfilter.filter-seeds` property.

By default, the seeds are read in as-is (operating under the assumption that the seed lists are hand-written, small, and easily managed by the user). However, there are some use cases where the seeds are extracted from a database and the user expects filtering behavior on a large list of seeds.

To filter the seed list:

1. In a text editor, open `default.xml`.
2. Set the `urlfilter.filter-seeds` property to `true`.
3. Save and close the file.

**Related Links**

*URL normalization properties* on page 20

You can set the URL normalization properties in the `default.xml` file.

*Normalizing the seed list* on page 21

You can apply normalization to the seed list with the `urlnormalizer.normalize-seeds` property.

## Crawl scoping properties

You can implement crawl scoping to control which URLs are crawled.

A crawl scope defines the conditions under which a URL is considered within the scope of a crawl. A URL is within the crawl scope if it should be fetched for that crawl.

Crawl scoping is applied before all other filters including the regular expressions in the `crawl-urlfilter.txt` file and custom plugins. This order of URL filtering means that even if a URL makes it through the crawl scope filter, it may still be filtered out by the `crawl-urlfilter.txt` file. However, a URL that is excluded by the crawl scope filter cannot be added by the `crawl-urlfilter.txt` file.

The crawl scope properties are listed in the following table.

| Property Name | Property Value |
|---|---|
| `crawlscope.mode` | `ANY`, `SAME_DOMAIN`, or `SAME_HOST` (default is `SAME_HOST`). Specifies the mode for crawl scoping. |
| `crawlscope.on-redirected-seed` | Boolean value (default is `true`). Specifies whether to filter a URL based on its seed or its redirected seed. |
| `crawlscope.top-level-domains.generic` | Space-delimited list of top-level domain names. Do not modify this list because it may affect how domain names are retrieved. Contains a list of generic top-level domain names. |
| `crawlscope.top-level-domains.additional` | Space-delimited list of top-level domain names (default is empty). Specifies additional top-level domain names that are pertinent to your crawls. |

### About configuring crawl scoping

The Web Crawler implements a basic crawl scoping scheme to accommodate crawls of multiple seeds. The crawler can scope a crawl to only visit URLs from the same host or from the same domain as a seed.

Crawl scoping is implemented via these properties:

- `crawlscope.mode`
- `crawlscope.on-redirected-seed`
- `crawlscope.top-level-domains.generic`
- `crawlscope.top-level-domains.additional`

The setting of the `crawlscope.mode` property determines the crawl scoping mode (that is, how URLs are allowed to be visited). The property sets one of these modes:

- `ANY` indicates that any URL is allowed to be visited. This mode turns off crawl scoping because there is no restriction on which URLs can be visited.
- `SAME_DOMAIN` indicates that a URL is allowed to be visited only if it comes from the same domain as the seed URL. The crawler attempts to figure out the domain name from examining the host.
- `SAME_HOST` (the default) indicates that a URL is allowed to be visited only if it comes from the same host as the seed URL.

The boolean setting of the `crawlscope.on-redirected-seed` property affects how redirections are handled when they result from visiting a seed. The property determines whether crawl scope filtering is applied to the redirected seed or to the original seed:

- `true` (the default) specifies that `SAME_HOST`/`SAME_DOMAIN` analysis will be performed on the redirected seed rather than the original seed.
- `false` specifies that `SAME_HOST`/`SAME_DOMAIN` filtering will be applied to the original seed.

Note that this redirect filtering property applies only to the `SAME_HOST` and `SAME_DOMAIN` crawl scope modes.

As an example of how these properties work, suppose the seed is set to `http://xyz.com` and a redirect is made to `http://xyz.go.com`. If the crawl is using `SAME_HOST` mode and has the `crawl.scope.on-redirected-seed` property set to `true`, then all URLs that are linked from here are filtered against `http://xyz.go.com`. If the redirect property is set to `false`, then all URLs that are linked from here are filtered against `http://xyz.com`.

The two `crawlscope.top-level-domains` properties are used for parsing domain names.

## How domain names are retrieved from URLs

Every domain name ends in a top-level domain (TLD) name. The TLDs are either generic names (such as `com`) or country codes (such as `jp` for Japan).

However, some domain names use a two-term TLD, which complicates the retrieval of top-level domain names from URLs.

For example:

- `http://www.xyz.com` has a one-term TLD of `com` with a domain name of `xyz.com`.
- `http://www.xyz.co.uk` has a two-term TLD of `.co.uk` with a domain name of `xyz.co.uk`

As the example shows, it is often difficult to generalize whether to take the last term or the last two terms as the TLD name for the domain name. If you take only the last term as the TLD, then it would work for `xyz.com` but not for `xyz.co.uk` (because it would incorrectly result in `co.uk` as the domain name). Therefore, the crawler must take this into account when parsing a URL for a domain name.

The two `crawlscope.top-level-domains` properties are used for determining which TLDs to use in the domain name:

- The `crawlscope.top-level-domains.generic` property contains a space-delimited list of generic TLD names, such as `com`, `gov`, or `org`.
- The `crawlscope.top-level-domains.additional` property contains a space-delimited list of additional TLD names that may be encountered in a crawl. These are typically two-term TLDs, such as `co.uk` or `ma.us`. However, you should also add country codes as necessary (for example, add `ca` if you are crawling the `www.xyz.ca` site). You should add TLDs to this list that are not generic TLDs but that you want to crawl.

The Web Crawler uses the property values as follows when retrieving domain names from URLs:

1. The crawler first looks at the last term of the host name. If it is a TLD in the `crawlscope.top-level-domains.generic` list (such as `com`), then the crawler takes the last two terms (`xyz` and `com`) as the domain name. This results in a domain name of `xyz.com` for the `http://www.xyz.com` sample URL.
2. If the last term is not one of the generic TLDs, then the crawler does the following: Takes the entire host name and checks it against the `crawlscope.top-level-domains.additional` list; if not a match, repeats by truncating the first term from the host name and checks it against the list; if not a match, repeats until a match is found or there are no more terms to be truncated from the host name.
3. If no terms matched on the `additional` list, return the last two terms as the domain name and log an error message.

For example, assume that you will be crawling `http://www.xyz.co.uk` and therefore want a domain name of `xyz.co.uk`. First you would add `co.uk` to the `crawlscope.top-level-domains.additional` list. The procedure for returning the domain name is as follows:

1. The generic TLD list is checked for the `uk` term, but it is not found.
2. `www.xyz.co.uk` is checked against the `crawlscope.top-level-domains.additional` list, but no match is found.
3. `xyz.co.uk` is checked against the additional TLD list, but no match is found.
4. `co.uk` is checked against the additional TLD list, and a match is finally found. A domain name of `xyz.co.uk` is returned.

If after step 4 no match is found in the `additional` list, the last two terms that were checked are returned as the domain name (`co.uk` in this example). In addition, a DEBUG-level message similar to this example is logged:

```
Failed to get the domain name for url: url
using result as the default domain name
```

where *url* is the original URL from which the domain name is to be extracted and *result* is a domain name consisting of the final two terms to be checked (such as `co.uk`). If you see this message, add the two terms to the `additional` list and retry the crawl.

## Default top-level domain names

The `crawlscope.top-level-domains.generic` property contains these TLD names in the `default.xml` configuration file that is shipped with the product:

- aero
- asia
- biz
- cat
- com
- coop
- edu
- gov
- info
- int
- jobs
- mil
- mobi
- museum

- name
- net
- org
- pro
- tel
- travel

As mentioned in the property table above, you should not modify this list because it may affect how domain names are determined.

## Document conversion properties

You can set the document conversion properties in the `default.xml` file.

The Endeca Web Crawler uses the CAS Document Conversion Module to perform text extraction on any document that is not one of these file types: HTML, text-based, or JavaScript. By using the properties listed in the table, you can configure the behavior of this module.

Note that the CAS Document Conversion Module respects the no-copy option of a PDF. That is, if a PDF publishing application has a no-copy option (which prohibits the copying or extraction of text within the PDF), the CAS Document Conversion Module does not extract text from that PDF. To extract the text, you must re-create the PDF without setting the no-copy option.

| Property Name | Property Value |
|---|---|
| `doc-conversion.attempts.max` | Integer value (default is 2). Specifies the maximum number of times that the module attempts to convert a document. |
| `doc-conversion.timeout` | Integer value (default is `60000`). Specifies the time-out value in milliseconds for converting a document. |

### Large files and the download content limit

Keep in mind that the `http.content.limit` property limits the maximum size of the content that can be downloaded. If the content is larger than the limit (an integer greater than 0), any content longer than the setting will be not be downloaded and you will see a WARN message similar to this example:

```
WARN com.endeca.itl.web.UrlProcessor
Content limit exceeded for http://xyz.com/pdf/B2B_info.pdf. Page will be
skipped.
```

This problem often occurs with large PDF files. If you constantly see these messages, increase the setting for the `http.content.limit` property.

**Related Links**

*HTTP Properties* on page 12

You can set the HTTP properties in the `default.xml` file.

## Output properties

You can set output properties in the `default.xml` file. You can configure output to either an output file (the default) or to a Record Store instance.

The properties in the table below allow you to specify the attributes of a crawl output file, such as its name, location, and output type. The default name of the output file is `endecaOut` and it is a compressed binary file by default.

**Note:** By default, the Web Crawler writes output to a file on disk. If desired, you can configure the Web Crawler to write output to a Record Store instance. Oracle recommends this approach.

| Property Name | Description |
|---|---|
| `output.file.directory` | Directory name (default is `workspace`). Specifies the directory for the output file. The name is case-sensitive and is relative to where you run the crawl from. You can specify a multi-level path. Note that this setting can be overridden with the `-w` command-line flag. |
| `output.file.name` | File name (default is `webcrawler-output`). Specifies the filename of the output file. The name is case-sensitive. |
| `output.file.is-xml` | Boolean value (default is `false`). Specifies whether the output type is XML (`true`) or binary (`false`). XML is useful if you want to visually inspect the Endeca records after crawling. |
| `output.file.is- compressed` | Boolean value (default is `true`). Specifies whether to compress the Endeca records in a .gz file. Setting this property to `true` is useful when storing and transferring large files. |
| `output.file.binary.file-size- max` | Integer value (default is `-1`). Sets the maximum file size for binary output files. Output is written to a new file once the maximum size is reached. If the value is set to `-1`, no limits are imposed on the file size. |
| `output.dom.include` | Boolean value (default is `false`). Specifies whether to include the DOM for the Web page in the output Endeca records. |
| `output.records.properties.excludes` | Space-delimited list of output record properties (default is empty). Specifies the properties that should be excluded from the records. The names can be specified in a case-insensitive format. Note that wildcard names are not supported. |
| `log.interval` | Integer value in seconds (default is `60`). Outputs crawl metrics information to the log every time this number of seconds has elapsed, per depth. |
| `log.interval.summary` | Integer value in seconds (default is `300`). Outputs detailed crawl progress information (organized by host) every time this number of seconds has elapsed. |

**Related Links**

*Configuring Web crawls to write output to a Record Store instance* on page 44

The Web Crawler can be configured to write its output directly to a Record Store instance, instead of to an output file on disk (the default). This procedure assumes you are modifying a single crawl configuration in the `site.xml` file and not the global Web crawler configuration in `default.xml`.

## Gathering XHTML information

If the `output.dom.include` property is set to `true`, the Web Crawler normalizes the content of HTML documents into XHTML and stores it in the `Endeca.Document.XHTML` property in the record.

1. In a text editor, open `default.xml`.
2. Set the `output.dom.include` to `true`.
   You can now extract information from the XHTML using XSLT or any other XML processing system.
3. Note that the `Endeca.Document.Text` property will also have extracted text, except that the XML header and the HTML tags are removed. Therefore, if you do not need the XHTML version of the content, set the `output.dom.include` property to `false`.
4. Save and close the file.

## Excluding record properties

The `output.records.properties.excludes` property allows you to specify a list of record properties that you want excluded from the records.

The list of the excluded property names is space delimited.

**Note:** Wildcards are not supported for the property names.

1. In a text editor, open `default.xml`.
2. Within the `<configuration>` element, add the following lines of code:

   ```
   <property>
      <name>output.records.properties.excludes</name>
      <value>excludedProperties</value>
   </property>
   ```

   Where *excludedProperties* is a space delimited list of the properties you wish to exclude.
3. Save and close the file.

### Example of excluding record properties

For example, assume you want to exclude both Outlink properties from the output. You would add this entry to the `site.xml` configuration file:

```
<property>
   <name>output.records.properties.excludes</name>
   <value>Endeca.Document.Outlink Endeca.Document.OutlinkCount</value>
</property>
```

On the next crawl, the `Endeca.Document.Outlink` and the `Endeca.Document.OutlinkCount` properties will not appear in the output.

**Note:** You can add the exclusion list to the `default.xml` file, but the `site.xml` file is recommended because you can then specify different property exclusions for different crawl configurations.

## Extensions for additional binary output files

For the `output.file.binary.file-size-max` property, if output has to be written to more than one output, the name pattern of the new files is similar to this example:

```
endecaOut-sgmt000.bin
endecaOut-sgmt001.bin
endecaOut-sgmt002.bin
```

That is, if the `output.file.name` value is set to `endecaOut`, then the suffix `-sgmt000` is used for the first file and the number is increased for subsequent files.

## Output file overrides in site.xml files

The `site.xml` files in the `workspace/conf/web-crawler/polite-crawl` and `workspace/conf/web-crawler/non-polite-crawl` directories contain these output file overrides.

| config property | default.xml | polite site.xml | non-polite site.xml |
|---|---|---|---|
| `output.file.directory` | `workspace` | `polite-crawl-workspace` | `non-polite-crawl-workspace` |
| `output.file.name` | `webcrawler-output` | `polite-crawl` | `non-polite-crawl` |
| `output.file.is-xml` | `false` | `true` | `true` |
| `output.file.is-compressed` | `true` | `false` | `false` |

# The site.xml file

The `site.xml` file provides override property values for the global configuration file.

The `default.xml` file is the global default configuration for your Endeca Web Crawler and should not change often. Only one copy of this file is shipped with the product, and it is located in the `workspace/conf/web-crawler/default` directory.

The `site.xml` file is where you make the changes that override the default settings on a per-crawl basis. The properties that you can add to the `site.xml` file are the same ones that are in the `default.xml` file. A `site.xml` file is included in the `workspace/conf/web-crawler/polite-crawl` and `workspace/conf/web-crawler/non-polite-crawl` directories, but not in the `workspace/conf/web-crawler/default` directory.

### Strategy for using the site.xml file

The strategy for using these two configuration files is to have only one directory that contains the `default.xml` file, but not a `site.xml` file. This directory is the default configuration directory.

You then create a separate directory for each different crawl-specific configuration. Each of these per-crawl directories will not contain the `default.xml` file, but will contain a `site.xml` file that is customized for a given crawl configuration.

When you run a crawl, you point to that crawl's configuration directory by using the `-c` command-line option. However, the Web Crawler is hard-coded to first read the configuration files in the `workspace/conf/web-crawler/default` directory and then those in the per-crawl directory (which can override the default files). For this reason, it is important that you do not change the name and location of the `workspace/conf/web-crawler/default` directory nor the `default.xml` file.

### Differences among the site.xml and default.xml files

The following table lists the differences between the `site.xml` files in the `non-polite-crawl` and the `polite-crawl` directories, as well as the differences between those files and the global `default.xml` file.

| config property | default.xml | polite site.xml | non-polite site.xml |
|---|---|---|---|
| `http.robots.ignore` | `false` | `false` | `true` |
| `fetcher.delay` | `2.0` | `1.0` | `0.0` |
| `fetcher.threads.total` | `100` | not used | `52` |
| `fetcher.threads.per-host` | `1` | `1` | `52` |
| `output.file.directory` | `workspace` | `polite-crawl-workspace` | `non-polite-crawl-workspace` |
| `output.file.name` | `webcrawler-output` | `polite-crawl` | `non-polite-crawl` |
| `output.file.is-xml` | `false` | `true` | `true` |
| `output.file.is-compressed` | `true` | `false` | `false` |

# The crawl-urlfilter.txt file

The `crawl-urlfilter.txt` file provides include and exclude regular expressions for URLs.

The `crawl-urlfilter.txt` file contains a list of include and exclude regular expressions for URLs. These expressions determine which URLs the crawler is allowed to visit. Note that the include/exclude expressions do not apply to seeds if `urlfilter.filter-seeds` is set to false.

Each regular expression must be prefixed by a + (plus) character or a - (minus) character. Plus-prefixed expressions are include expressions while minus-prefixed expressions are exclude expressions.

Note that the name of this file is specified to the Web Crawler via the `urlfilter.regex.file` property in the `default.xml` configuration file.

## Regular expression format

The Web Crawler implements Sun's java.util.regex package to parse and match the pattern of the regular expression. Therefore, the supported regular-expression constructs are the same as those in the documentation page for the java.util.regex.Pattern class:

```
http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html
```

This means that among the valid constructs you can use are:

- Escape characters, such **\t** for the tab character.
- Character classes (simple, negation, range, intersection, subtraction). For example, **[^abc]** means match any character except a, b, or c, while **[a-zA-Z]** means match any upper- or lower-case letter.
- Predefined character classes, such as **\d** for a digit or **\s** for a whitespace character.
- POSIX character classes (US-ASCII only), such as **\p{Alpha}** for an alphabetic character, **\p{Alnum}** for an alphanumeric character, and **\p{Punct}** for punctuation.
- Boundary matchers, such as **^** for the beginning of a line, **$** for the end of a line, and **\b** for a word boundary.

- Logical operators, such as **X|Y** for either X or Y.

For a full list of valid constructs, see the Pattern class documentation page referenced above.

## Specifying the hosts to accept

You can set the `crawl-urlfilter.txt` files to accept certain hosts.

The `crawl-urlfilter.txt` files in the configuration directories (`default`, `polite`, and `non-polite`) all have this line commented out:

```
# accept hosts in MY.DOMAIN.NAME
# +^http://([a-z0-9]*\.)*MY.DOMAIN.NAME.com/
```

To limit the crawl to a specific domain:

1. In a text editor, open `crawl-urlfilter.txt`.
2. Replace "MY.DOMAIN.NAME" with the domain name that you are crawling, and make this a non-comment line.
3. At the end of the file, replace the plus sign with a minus sign and update the comment as follows:

   ```
   # exclude everything else
   -.
   ```

4. Save and close the file.

**Example of specifying hosts to accept**

Specify the hosts to accept in these lines:

```
# accept hosts within endeca.com
+^http://([a-z0-9]*\.)*endeca.com/
```

Then change the last lines of the file:

```
# include everything
+.
```

to replace the plus sign with a minus sign:

```
# exclude everything else
-.
```

With these two changes, hosts within the `endeca.com` domain will be accepted by the crawler and everything else will be excluded.

## Order of the regular expressions

When entering regular expressions, make sure that you enter the exclude expressions before the include expressions. The reason is that the RegexURLFilter plugin does the regex-pattern matching from top to bottom.

This means that if there is a match, then that match takes precedence. Therefore, if you have the include pattern first, then the exclude patterns following it would not take effect.

For example, assume that you have these two entries:

```
+^http://mysite.com/public
-^http://mysite.com/public/oldcontent
```

In this case, the `oldcontent` exclusion will never take effect because the `public` matching takes precedence.

## Excluding file formats

You can globally exclude file formats by adding their file extensions to an exclusion line in the `crawl-urlfilter.txt` file.

The default `crawl-urlfilter.txt` configuration excludes these file types:

- BMP (bitmap image), via the .bmp and .BMP extensions
- CSS (Cascading Style Sheet), via the .css extension
- EPS (Encapsulated PostScript), via the .eps extension
- EXE (Windows executable), via the .exe extension
- GIF (Graphics Interchange Format), via the .gif and .GIF extension
- GZIP (GNU Zip), via the .gz extension
- ICO (icon image), via the .ico and .ICO extension
- JPG and JPEG (Joint Photographic Experts Group), via the .jpeg, .JPEG, .jpg, and .JPG extensions
- MOV (Apple QuickTime Movie), via the .mov and .MOV extensions
- MPG (Moving Picture Experts Group), via the .mpg extension
- PNG (Portable Network Graphics), via the .png and .PNG extension
- RPM (Red Hat Package Manager), via the .rpm extension
- SIT ( Stuffit archive), via the .sit extension
- TGZ (Gzipped Tar), via the .tgz extension
- WMF (Windows Metafile), via the .wmf extension
- ZIP (compressed archive), via the .zip extension

Except for HTML, text-based, and JavaScript files, text conversion on all other file types is performed by the CAS Document Conversion Module (if you have installed and enabled the module). As a rule of thumb, therefore, you should exclude any file format that is not supported by the module. For a list of the supported file formats, see the *CAS Developer's Guide*.

1. To exclude file formats:
2. In a text editor, open `crawl-urlfilter.txt`.
3. Locate the following lines:

   ```
   # skip image and other suffixes we can't yet parse
   -\.(gif|GIF|jpg|JPG|...|bmp|BMP)$
   ```

   (the example is truncated for ease of reading)
4. Modify the second line to reflect file extensions that you wish to exclude.
5. Save and close the file.

# The regex-normalize.xml file

The `regex-normalize.xml` file provides substitutions for normalizing URLs.

The `regex-normalize.xml` file is the configuration file for the RegexUrlNormalizer class. The file allows you to specify regular expressions that can be used as substitutions for URL normalization. The file provides a set of rules as sample regular expressions.

For example, if you are crawling a site with URLs that contain spaces, you should add the following regular expression to force URL encoding:

```
<regex>
    <pattern> </pattern>
     <substitution>%20</substitution>
<regex>
```

Note that the expression uses one space character as the value for the pattern. The expression means that when a space character is found in the URL, the space should be encoded as %20 (hex). For example, if the URL contains a document named `Price List.html`, it will be encoded to `Price%20List.html` so that it can be processed correctly.

When modifying the file, keep the following in mind:

- The rules are applied to URLs in the order that they occur in the file.
- Because an XML parser reads the file, ampersand (&) characters must be expanded to their HTML equivalent (&amp;).

Note that the name of this file is specified to the Web Crawler via the `urlnormalizer.regex.file` property in the `default.xml` configuration file.

# The mime-types.xml file

The `mime-types.xml` file provides mappings of file extensions to MIME types.

The `mime-types.xml` file provides definitions of MIME types by associating file extensions with the names of MIME types and providing magic sequences.

Note that the name of this file is specified to the Web Crawler via the `mime.types.file` property in the `default.xml` configuration file.

# The parse-plugins.xml file

The `parse-plugins.xml` file provides mappings of MIME types to parsers.

The `mime-types.xml` file has two purposes:

- It maps MIME types to parsers, that is, which parsing plugin should be called for a particular MIME type. For example, it maps the HtmlParser to the `text/html` MIME type.
- It provides the order in which plugins are invoked for the MIME types.

Note that the name of this file is specified to the Web Crawler via the `parse.plugin.file` property in the `default.xml` configuration file.

This entry from the file shows how these parsing rules are set:

```
<mimeType name="text/xml">
   <plugin id="parse-html" />
   <plugin id="endeca-searchexport-converter-parser" />
</mimeType>
```

In this entry, the HtmlParser plugin is first invoked for a `text/xml` MIME type. If that plugin is successful, the parsing is finished. If it is unsuccessful, then the endeca-searchexport-converter-parser plugin is invoked.

Note that this entry:

```
<mimeType name="*">
   <plugin id="endeca-searchexport-converter-parser" />
</mimeType>
```

indicates that the endeca-searchexport-converter-parser plugin is invoked for any unmatched MIME type.

In general, you should not modify the contents of this file unless you have written your own parser plugin.

# The form-credentials.xml file

The `form-credentials.xml` file provides the credentials for sites that use form-based authentication.

Note that a template `form-credentials.xml` file is shipped in the `conf/web-crawler/default` directory. You can create a credentials file that corresponds to the needs of your crawl.

## About Form-based authentication

The Web Crawler supports Form-based authentication for both GET and POST requests. The `http.auth.form.credentials.file` property sets the name of the file that contains the form credentials to be used by the Web client.

If a Web server uses HTML forms to restrict access to Web sites, you can specify authentication credentials that enable the Web Crawler to access password-protected pages.

The fields that you specify in the credentials file correspond to the fields that an interactive user fills in when prompted by the Web browser, and any hidden or static fields that are required for a successful login. This means that you must coordinate with the server administrators, who must provide you with the security requirements for the Web sites, including all information that is used to authenticate the Web Crawler's identity and determine that the crawler has permission to crawl the restricted pages.

In the Web Crawler, the authentication plugin provides a way to execute form-based login for Web crawls. The plugin implements two main authentication modes:

- Pre-crawl authentication mode performs the authentication before the crawl begins. Note that if pre-crawl authentication is specified and the request times out, the Authenticator will attempt an in-crawl authentication for the retry.
- In-crawl authentication mode performs the authentication as the crawl is progressing. After every page is fetched and processed, a site-specific authenticator checks the page contents and determines whether or not the page needs to be refetched (say, if the crawler has been logged out), and it may log into the site if necessary.

The `preCrawlAuth` setting in the credentials file determines whether pre-crawl or in-crawl authentication is performed. If you are uncertain as which mode to use, we recommend that you start by using the pre-crawl mode, as long as you think that the authentication process will not time out. If, however, you believe that timeouts will occur, then the in-crawl mode would be more advantageous.

## Format of the credentials file

The format of the form-based authentication credentials file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<credentials>
   <formCredentials>
      <authenticator>
         <className>authClass</className>
         <configuration>
            <siteUrlPattern>siteUrl</siteUrlPattern>
            <loginUrl>loginPageUrl</loginUrl>
            <actionUrl>actionUrl</actionUrl>
            <method>authMethodToUse</method>
            <preCrawlAuth>shouldPreAuth</preCrawlAuth>
            <parameters>
               <parameter>
                  <name>paramName</name>
                  <value>paramValue</value>
               </parameter>
            </parameters>
            <properties>
               <property>
                  <name>propName</name>
                  <value>propValue</value>
               </property>
            </properties>
         </configuration>
      </authenticator>
   </formCredentials>
</credentials>
```

The meanings of the elements and attribute values are listed in the following table.

| Element | Meaning |
|---|---|
| `<credentials>` and `<formCredentials>` | Main opening elements. There can be only one set of these elements in the file. |
| `<authenticator>` | Defines one set of settings for the Authenticator plugin. The file will have multiple `<authenticator>` sections if the site has multi-form authentication. |
| `<className>` | The name of the class that handles authentication logic. The Web Crawler default authenticator class is: `com.endeca.itl.web.auth.form.BasicFormAuthenticator`. If desired, you can override this class with a custom authentication class you that implement. |
| `<configuration>` | Defines a set of credentials settings and properties. |
| `<siteUrlPattern>` | A regular expression that determines which sites will be authenticated (i.e., the Authenticator will be run only on those sites). |
| `<loginUrl>` | The URL where the actual login is done (such as `http://samplesite.com/login.html`). |
| `<actionUrl>` | A full path to a URL that handles the logic for the GET/POST request, such as a CGI script. This field corresponds to the ACTION attribute of the form. Note that an action URL is often different from the login URL. |

| Element | Meaning |
|---|---|
| `<method>` | A value of either `GET` or `POST`. |
| `<preCrawlAuth>` | Boolean value. Indicates whether authentication is done before the crawl starts (a value of `true` enables pre-crawl authentication) or whether the authentication is done during the crawl (a value of `false` enables in-crawl authentication ). |
| `<parameters>` | Contains one or more sets of `<parameter>` elements. The parameters correspond to the form fields you wish to fill out (such as the login name and password). By default, the parameters are all included with the HttpRequest sent to the server. |
| `<parameter>` | Contains a `<name>` element that is the name of a field in the form and a `<value>` element that is the value to be supplied for that field. |
| `<properties>` | Contains one or more sets of `<property>` elements. They are placed in the Property map and can be accessed as Strings. Properties are meant to be specific settings for the Authenticator plugin, and allow a way for the plugin to be customized easily. Note that this element is optional. |
| `<property>` | Contains a `<name>` element that is the name of a property and a `<value>` element that is the value of that property. |

## Setting the timeout property

You can set the authentication timeout with the BasicFormAuthenticator.

The `timeout` property specifies the logout expiration in milliseconds. If this property is not specified, it sets the timeout to be the default of `-1` (infinite, i.e., no logout expiration).

To set the `timeout` property:

1. In a text editor, open the `form-credentials.xml` file.
2. Locate the `timeout` property.
3. Modify the property's value as needed.
4. Save and close the file.

## Using special characters in the credentials file

XML has a special set of characters that cannot be used in normal XML strings. If you need to enter any of the following special characters, you must enter them in their encoded format:

| Special Character | Encoded Format |
|---|---|
| & | &amp; |
| < | &lt; |
| > | &gt; |
| ' | &apos; |
| " | &quot; |

For example, if the string `he&l>lo` is the login password, then the credentials file would have this entry:

```
<parameter>
   <name>PASSWORD</name>
   <value>he&amp;l&gt;lo</name>
</parameter>
```

## Authentication Exceptions

The authentication framework has two Exception classes:

- An `AuthenticationFailedException` is thrown if an error prevents the authentication (for example, the password is wrong).
- A `RequestFailedException` is thrown if a non-authentication error occurs (for example, the HTTP connection suddenly shuts down).

# The log4j.properties file

The `log4j.properties` file sets the logging properties.

You can modify the `log4j.properties` file to change the properties for the log4j loggers.

### Default log4j properties

The default `log4j.properties` file has this configuration:

```
log4j.rootLogger=ERROR,stdout
log4j.logger.com.endeca=INFO
# Logger for crawl metrics
log4j.logger.com.endeca.itl.web.metrics=INFO

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPat¬
tern=%p\t%d{ISO8601}\t%r\t%c\t[%t]\t%m%n
```

The presence of only the `ConsoleAppender` means that the standard output is directed to the console, not to a log file.

### Logging to a file

You can change the default `log4j.properties` configuration so that messages are logged only to a file or to both the console and a file. For example, you would change the above configuration to a configuration similar to this:

```
# initialize root logger with level ERROR for stdout and fout
log4j.rootLogger=ERROR,stdout,fout
# set the log level for these components
log4j.logger.com.endeca=INFO
log4j.logger.com.endeca.itl.web.metrics=INFO

# add a ConsoleAppender to the logger stdout to write to the console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
# use a simple message format
log4j.appender.stdout.layout.ConversionPattern=%m%n
```

```
# add a FileAppender to the logger fout
log4j.appender.fout=org.apache.log4j.FileAppender
# create a log file
log4j.appender.fout.File=crawl.log
log4j.appender.fout.layout=org.apache.log4j.PatternLayout
# use a more detailed message pattern
log4j.appender.fout.layout.ConversionPat¬
tern=%p\t%d{ISO8601}\t%r\t%c\t[%t]\t%m%n
```

In the example, the `FileAppender` appends log events to the log file named `crawl.log` (which is created in the current working directory). The `ConsoleAppender` writes to the console using a simple pattern in which only the messages are printed, but not the more verbose information (logging level, timestamp, and so on).

In addition, you can change the component logging levels to any of these:

- `DEBUG` designates fine-grained informational events that are most useful to debug a crawl configuration.
- `TRACE` designates fine-grained informational events than `DEBUG`.
- `ERROR` designates error events that might still allow the crawler to continue running.
- `FATAL` designates very severe error events that will presumably lead the crawler to abort.
- `INFO` designates informational messages that highlight the progress of the crawl at a coarse-grained level.
- `OFF` has the highest possible rank and is intended to turn off logging.
- `WARN` designates potentially harmful situations.

These levels allow you to monitor events of interest at the appropriate granularity without being overwhelmed by messages that are not relevant. When you are initially setting up your crawl configuration, you might want to use the `DEBUG` level to get all messages, and change to a less verbose level in production.

Note the default `log4j.properties` file contains a number of suggested component loggers that are commented out. To use any of these loggers, remove the comment (#) character.

# Enabling the CAS Document Conversion Module with the Web Crawler

By default, the Web Crawler is configured to call the CAS Document Conversion Module to convert any documents that are not text, HTML, SGML, or JavaScript.

# Disabling the CAS Document Conversion Module with the Web Crawler

If desired, you can disable the CAS Document Conversion Module to prevent document conversion or license warnings. You can either disable the module globally for all crawls, or you can disable the module on a per crawl basis.

1. To change the default setting for all crawls:
   a) Navigate to `<install path>\CAS\workspace\conf\web-crawler\default`.

b) In a text editor, open `default.xml`.

c) Add a property named `plugin.excludes` and specify a value of `endeca-searchexport-converter-parser`.
For example:

```
<property>
      <name>plugin.excludes</name>
      <value>endeca-searchexport-converter-parser</value>
      <description>Disable the CAS Document Conversion Module from
running.
      </description>
</property>
```

d) Save and close the file.

2. To change the setting on a per crawl basis:

a) Navigate to `<install path>\CAS\workspace\conf\web-crawler\<crawl name>`.

b) In a text editor, open `site.xml`.

c) Add a property named `plugin.excludes` and specify a value of `endeca-searchexport-converter-parser`.
For example:

```
<property>
      <name>plugin.excludes</name>
      <value>endeca-searchexport-converter-parser</value>
      <description>Disable the CAS Document Conversion Module from
running as part of this crawl configuration.
      </description>
</property>
```

d) Save and close the file.

## About Document Conversion options

You can change the default behavior of the CAS Document Conversion Module by specifying options via JVM property names and values.

Note that you cannot set these options via the standard configuation files.

The two options are:

- `stellent.fallbackFormat` determines the fallback format, that is, what extraction format will be used if the CAS Document Conversion Module cannot identify the format of a file. The two valid settings are `ascii8` (files whose types are specifically unidentifiable are treated as plain-text files, even if they are not plain-text) and `none` (unrecognized file types are considered to be unsupported types and therefore are not converted). Use the `none` setting if you are more concerned with preventing many binary and unrecognized files from being incorrectly identified as text. If there are documents that are not being properly extracted (especially text files containing multi-byte character encodings), it may be useful to try the `ascii8` option.
- `stellent.fileId` determines the file identification behavior. The two valid settings are `normal` (standard file identification behavior occurs) and `extended` (an extended test is run on all files that are not identified). The `extended` setting may result in slower crawls than with the `normal` setting, but it improves the accuracy of file identification.

### Default values for the options

The default values are as follows:

| Option | Defaults value |
|---|---|
| `stellent.fallbackFormat` | none |
| `stellent.fileId` | extended |

## Setting document conversion options

Set the document conversion options as parameters to the Java Virtual Machine (JVM), via the Java `-D` option.

To set the fallback format, use one of these two parameters:

1. Run the startup script with the `-JVM` flag.

   **Note:** When using the `-JVM` flag, it must be the last flag on the command line.

2. Set the fallback format using one of these two parameters:
   - `-Dstellent.fallbackFormat=ascii8`
   - `-Dstellent.fallbackFormat=none`

3. Set the file identification behavior using one of these two parameters:
   - `-Dstellent.fileId=normal`
   - `-Dstellent.fileId=extended`

### Example of setting document conversion options

```
.\bin\web-crawler -d 2 -s mysites.lst -JVM "-Dstellent.fallbackFormat=ascii8"
```

**Note:** On Windows machines, the parameters should be quoted if they contain equals signs.

# Configuring Web crawls to write output to a Record Store instance

The Web Crawler can be configured to write its output directly to a Record Store instance, instead of to an output file on disk (the default). This procedure assumes you are modifying a single crawl configuration in the `site.xml` file and not the global Web crawler configuration in `default.xml`.

There are two main tasks in the configuration process. You create and configure a Record Store instance to receive the Web Crawler output. Then you configure the Web Crawler to override its default output settings and instead write to the Record Store instance.

The Record Store instance configuration requires a configuration file with two properties for Web Crawler output. The Web Crawler configuration requires the following two changes to the `site.xml` file:

- Add three output properties to specify the host and port of the machine running the Record Store, and instance name of the Record Store that you want to write to.
- Add a `plugin.includes` property for the **recordstore-outputter** plugin. This plugin instructs the Web Crawler to write to a Record Store instance and over rides the **output-endeca-record** which would have instructed the Web Crawler to write to an output file.

Each of these steps is fully described below.

To configure a Web Crawler to write output to a Record Store instance:

1. Start the Endeca CAS Service if it is not running already

   On Windows, the Endeca CAS Service is started by default.

2. Using the Component Instance Manager Command-line Utility, create a new Record Store instance for the Web Crawler output.

   a) Start a command prompt and navigate to `<install path>\CAS\`*`version`*`\bin`.

   b) Run the `create-component` task of `component-manager-cmd`. Specify the `-t` option with an argument of `RecordStore`. Specify the `-n` option with a Record Store instance name of your choice. If necessary, specify host and port information or accept the defaults.
   For example, this Windows command creates a Record Store instance named `WebCrawlOutput`:

   ```
   C:\Endeca\CAS\3.0.2\bin>component-manager-cmd.bat create-component
   -h localhost -n WebCrawlerOutput -p 8500 -t RecordStore
   ```

   The command prompt displays:

   ```
   Successfully created component: WebCrawlerOutput
   ```

3. Create a Record Store configuration file that has an `idPropertyName` property of `Endeca.Id` and `changePropertyNames` of `Endeca.Document.Text, Endeca.Web.Last-Modified`. For example, here are the contents of a configuration file named `recordstore-configuration.xml`:

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <recordStoreConfiguration xmlns="http://recordstore.itl.endeca.com/">
       <changePropertyNames>
        <changePropertyName>Endeca.Document.Text</changePropertyName>
        <changePropertyName>Endeca.Web.Last-Modified</changePropertyName>
       </changePropertyNames>
       <idPropertyName>Endeca.Id</idPropertyName>
   </recordStoreConfiguration>
   ```

4. Save the Record Store configuration file. You may find it convenient to save it with the other Web Crawler configuration files.

5. Using the Record Store Command-line Utility, set the configuration file for the Record Store instance.

   a) Start a command prompt and navigate to `<install path>\CAS\`*`version`*`\bin`.

   b) Run the `set-configuration` task of `recordstore-cmd`. Specify the `-a` option with an argument of the Record Store instance name. Specify the `-f` option with the path to the configuration file for the Record Store instance.
   For example, this Windows command sets the configuration file named `recordstore-configuration.xml` for the Record Store instance named `WebCrawlerOutput`:

   ```
   C:\Endeca\CAS\3.0.2\bin>recordstore-cmd.bat set-configuration
   -a WebCrawlerOutput -f C:\sample\webcrawler\recordstore-configuration.xml
   ```

   The command prompt displays:

   ```
   Successfully set recordstore configuration.
   ```

6. Modify the `site.xml` file to include the three output properties that specify the fully qualified name of the host and the port on which the Record Store is running and the instance name of the Record Store.
   For example, this snippet specifies an instance name of `WebCrawlerOutput` with defaults for a Record Store running locally:

   ```
   <property>
      <name>output.recordStore.host</name>
      <value>hostname.endeca.com</value>
   </property>
   <property>
      <name>output.recordStore.port</name>
      <value>8500</value>
   </property>
   <property>
      <name>output.recordStore.instanceName</name>
      <value>WebCrawlerOutput</value>
   </property>
   ```

7. In the `site.xml` file, add a `plugin.includes` property for the **recordstore-outputter** plugin. This plugin instructs the Web Crawler to write to a Record Store instance.
   For example:

   ```
   <property>
      <name>plugin.includes</name>
        <value>lib-auth-http|auth-http-form-basic|protocol-httpclient|proto¬
   col-file|urlfilter-regex|parse-(text|html|js)|endeca-searchexport-convert¬
   er-parser|urlnormalizer-(pass|regex|basic)|endeca-generator-html-ba¬
   sic|recordstore-outputter</value>
   </property>
   ```

8. In the `site.xml` file, delete the `plugin.includes` property for the **output-endeca-record** plugin, if it exists in the file.
9. Optionally, you can remove properties in `site.xml` file that configure output file settings. These properties include: `output.file.is-compressed`, `output.file.is-xml`, `output.file.name`, and `output.file.directory`.

   Removing them is useful if you want a clean configuration file, but removing them is not required because the addition of the **recordstore-outputter** plugin over rides the file output properties.
10. Run the Web crawl as you normally would.

To confirm the Web crawl wrote output to a Record Store instance, run the `list-generations` task of the Record Store Command-line Utility. For the example above, this command confirms the crawl output for the `WebCrawlerOutput` instance:

```
C:\Endeca\CAS\3.0.2\bin>recordstore-cmd list-generations -a WebCrawlerOutput
ID      STATUS          CREATION TIME
1       COMPLETED       Tue Mar 03 17:40:22 EST 2009
```

**Note:** The Web Crawler does not automatically manage Record Store instances for Web crawls. For details about managing Record Store instances, see the *CAS Developer's Guide*.

Chapter 3

# Supported crawl types

This section provides an overview of the full and resumable crawl types that are supported by the Endeca Web Crawler.

## About full crawls

This topic provides an overview of full crawls.

A *full* crawl means that the crawler processes all the pages in the seeds (except for pages that are excluded by filters). As part of the full crawl, a crawl history database is created with metadata information about the URLs. The database is created in the workspace directory of the crawl.

The crawl database provides persistence, so that its history can later be used for resumable crawls. For example, if the user stops a full crawl via a Control-C in the command window, the crawler closes the database files before exiting. If the crawl is later resumed (via the `-r` flag), the resumed crawl begins with the first URL that has a status of *pending*.

**Workflow of a crawl**

The Web Crawler handles full crawls as follows:

1. The crawler creates the crawl history database. If a previous database exists, it is overwritten.
2. The depth of the crawl is entered in the database.
3. From the seed, the crawler generates a list of URLs to be visited and queues them in the database. Each URL is given a status of *pending* because it has not yet been visited.
4. The crawler gets a URL from the queue, visits (and processes) the page, and changes the URL's status in the database to *complete*.
5. The crawler repeats step 4 until all the queued URLs are processed.

## About resumable crawls

This topic provides an overview of resumable crawls.

A *resumable* crawl (also called a *restartable* crawl) is a crawl that uses the seed URLs of a previous full or resumed crawl. It also uses a greater depth level and/or a different set of configuration settings.

You use the `-r` (or `--resume`) command-line flag to resume a crawl. Resumable crawls use the previously-created crawl history database in the workspace directory, because the database provides the seed and a list of URLs that have already been crawled. Resumable crawls do not recrawl URLs that have a status of *complete* in the history database.

Among the possible use-case scenarios for resumable crawls are the following:

- You have successfully run a crawl (for example, a test crawl using a depth of 0). Now you want to run the same crawl again (i.e., same seeds and same configuration), but this time with a greater depth. However, because you have the output from the first crawl, you do not want to recrawl those pages, but instead want to start from where the first crawl finished.
- You have successfully run a crawl, and now want to run the same crawl (i.e., same seeds) but with a different configuration. Again, you do not want to recrawl any previously-crawled pages and want to keep the output from the first crawl.

The rules for resumed crawls are the following:

- A previous crawl must have been successfully run. That is, the previous crawl must have generated a history (state) database that will be used as a starting point for the resumed crawl. Note that crawls that were stopped (e.g., via a Control-C in the command window) are considered successful crawls if the crawl was gracefully shut down (that is, the history database is up-to-date).
- The same seed must be used. That is, you cannot use the `-s` flag to specify a different seed for the resumed crawler (the flag is ignored if you use it). Instead, the Web Crawler will use the the seed from the history database. Because the history database also contains the list of URLs that were crawled, the resumed crawl will not recrawl those URLs.
- The same workspace directory must be used. You cannot use the `-w` flag to specify a different workspace directory. The reason is that the resumed crawl must use the same history database as the previous crawl (and must also update that database with the newly-crawled information).
- You must use the `-d` flag to a greater crawl depth than the previous crawl. If you specify a crawl depth that is less than or the same as the previous crawl, no records are generated. (However, if you have the same depth as the previous crawl and the previous crawl did not finish that depth, then records will be generated.) This same rule also applies to the maximum number of requests to be made (via the `-l` flag).
- The `-c` flag can be used to provide a different configuration for the resumed crawl. The new configuration is used for the uncrawled pages, but does not affect pages that have already been crawled.
- Because you can change the configuration, you can specify a new output file name.
- The `-f` flag cannot be used.

# About workspace directories and output files

This topic describes file output settings. By default, Web crawls use the workspace directory to store their output files. For details about Record Store settings, see the CAS Developer's Guide.

### Workspace directory

When a crawl is run, you specify its workspace directory either explicitly (via a path in the `-w` flag) or implicitly (via the `output.file.directory` property in the configuration file). Note that the `-w` flag overrides the setting of the `output.file.directory` property if the values are different.

By default, the workspace directory has these subdirectories:

- `output` – default location for the crawl output files.
- `state/web` – location of the crawl history database.
- `logs` – location of log files, such as `derby.log` for the crawl database.

If you are running simultaneous crawls, each crawl must have its own workspace directory.

### Record output file

The name of a crawl output file is set by the `output.file.name` property in the `default.xml` configuration file (which can be overridden by the `site.xml` file). Assuming the default name of `endecaOut`, the full name of the output file depends on the configuration settings :

- For compressed binary files (the default), `endecaOut-sgmt000.bin.gz` will be the name. If more than one output file is generated, the second file will be `endecaOut-sgmt001.bin.gz`, and so on.
- For uncompressed binary files, `endecaOut-sgmt000.bin` will be the name of the first file, `endecaOut-sgmt001.bin` for the second file, and so on.
- For XML files, the name will be either `endecaOut.xml.gz` (if compression is specified) or `endecaOut.xml` (if compression is turned off). Note that unlike the binary format, only one XML file is output, regardless of its size.

The format of the file is set with the `output.file.is-xml` property, while the `output.file.is-compressed` property turns compression on or off.

### Archived output files

For the first time that a crawl is run in a given workspace directory, the output file is named as described in the previous section. For example, if you run a full crawl, the output filename might be `endecaOut-sgmt000.bin.gz`. If you then run a second crawl (full or resumable), the Web Crawler works as follows:

1. A directory named `archive` is created under the `output` directory.
2. The original `endecaOut-sgmt000.bin.gz` file is moved to the `archive` directory and is renamed by adding a timestamp to the name; for example:

   ```
   endecaOut-20091015173554-sgmt000.bin.gz
   ```

3. The output file from the second run is named `endecaOut-sgmt000.bin.gz` and is stored in the `output` directory.
4. For every subsequent crawl using the same workspace directory, steps 2 and 3 are repeated.

The timestamp format used for renaming is:

```
YYYYMMDDHHmmSS
```

where:

- YYYY is a four-digit year, such as 2009.
- MM is the month as a number (01-12), such as 10 for October.
- DD is the day of the month, such as 15 (for October 15th).
- HH is the hour of the day in a 24-hour format (00-23), such as 17 (for 5 p.m.).
- mm is the minute of the hour (00-59).
- SS is the second of the minute (00-59).

Note that the timestamp format is hard-coded and cannot be reconfigured.

Chapter 4

# Running the Endeca Web Crawler

This section provides information on how to run the Endeca Web Crawler, including the startup scripts and the record properties that are returned by the crawls.

## Command-line flags for crawls

The Endeca Web Crawler startup script has several flags to control the behavior of the crawl.

The `web-crawler` startup script has the following flags . If used with no flags, the `web-crawler` script displays the usage information and exits.

| web-crawler Flag | Flag Argument |
|---|---|
| `-c` or `--conf` | The path of the configuration directory. If this flag is not specified, the `workspace/conf/web-crawler/default` directory is used as the default configuration directory. Note that if the flag points to a directory other than the default configuration directory (such as `workspace/conf/web-crawler/polite-crawl`), the files in the `workspace/conf/web-crawler/default` directory are read first. Optional. |
| `-d` or `--depth` | A non-negative integer (greater than or equal to `0`) that specifies the maximum depth of the crawl. The crawl depth is the number of link levels (starting from the seed URL) that will be crawled (see below for details). Required for all crawl types. |
| `-f` or `--force` | No argument to the flag. This flag forces the output directory to be deleted before the crawl is run. Optional, but it cannot be used with resumable crawls. |
| `-JVM` | Allows arguments on the command line to be passed to the Java Virtual Machine (JVM). If this flag is used, any arguments before it are passed to the Web Crawler and any arguments afterwards are appended to those passed to the JVM. Note that on Windows machines, the flag parameters should be quoted if they contain equal signs. Optional. |
| `-l` or `--limit` | An integer that specifies an approximate maximum for the number of requests to make (that is, the maximum number of pages to be fetched). The maximum is a soft limit: when the limit is reached, the Crawler does not add any more pages to the queue but the Crawler completes any pages still in the queue. |

| web-crawler Flag | Flag Argument |
|---|---|
| | When the limit is reached, the Web Crawler also writes a `URL limit reached, starting shutdown.` message to the log file.<br>The default is `0` (zero), which sets no limit. This flag is useful when you are setting up and testing your crawl configuration. Optional. |
| `-r` or `--resume` | Resumes a full or resumable crawl that has been previously run. Optional. |
| `-s` or `--seed` | The seed for the crawl. The seed can be one URL, a file containing URLs (one per line), or a directory containing *.lst files that contain URLs. The URLs must be fully qualified, not just the domain names; that is, you must specify the protocol (`http://` or `https://`) and, if the port is not 80, the port number. The default port of HTTP is 80. The default port for HTTPS is 443. Required for full crawls, but is ignored for resumable crawls . |
| `-w` or `--working` | The path of the Web Crawler workspace directory. If this flag is not used, the default name of the workspace directory is `workspace` and is located in the directory from which the startup script is run. Because each `workspace` directory must have a unique path, you must use this flag if you are starting multiple Web crawls on the same machine. Optional. |

### Setting the crawl depth

The crawl depth (as set by the `-d` flag) specifies how many levels of page links will be followed. Each URL in the seed has a level of 0 and each link from a seed URL has a level of 1. The links from a level 1 URL have a level of 2 and so on.

For example, if the seed is `www.endeca.com`, the levels are as follows:

```
Level 0: www.endeca.com is level 0 and has a link to about.html.
   Level 1: about.html is level 1 and its links are level 2.
      Level 2: contacts.html is level 2 and its links are level 3.
```

Therefore, if you want to crawl all the level 2 pages, specify `-d 2` as the flag argument.

### Specifying the configuration directory

The `workspace/conf/web-crawler/default` directory is the default configuration directory. For example, this directory is used if you do not specify the `-c` flag.

You can also use the `-c` flag to override one or more configuration files in the default configuration directory with files from another configuration directory. For example, assume you have a directory (named `intsites`) that has a `site.xml` file for a specific crawl (and no other configuration files). You would then use the `-c` flag to point to that directory:

```
.\bin\web-crawler -c conf\web\intsites -d 2 -s conf\web\intsites\int.lst
```

In this example, the crawl uses the `site.xml` from the `intsites` directory, while the rest of the files are read from the default configuration directory.

### Specifying JVM arguments

To pass additional arguments to the Java Virtual Machine (JVM), you can use the `-JVM` script flag. For example, assume you want to override the default maximum heap size setting of 1024 MB that is hardcoded in the scripts with a setting of 2048 MB. The command line might be as follows:

```
.\bin\web-crawler -d 2 -s conf\web\intsites\int.lst -JVM -Xmx2g
```

Keep in mind that this flag must be the last flag on the command line, because any arguments that follow it are appended to those passed to the JVM.

# Running full crawls

You run full crawls from the command line.

A full crawl means that the crawler processes all the URLs in the seed (except for URLs that are excluded by filters). By default, a crawl history database is created in the `workspace/state/web` directory.

You can run multiple, simultaneous crawls on the same machine. When running multiple crawls, each crawl must have its own workspace directory. All the crawls can use the same configuration, or they can use a crawl-specific configuration.

**Note:** If you are using the default configuration, you must run Web crawls from the Web Crawler root directory (i.e., the `CAS\`*`version`* directory). To run crawls from other directories, you must change the `plugin.folders` configuration property so that it uses an absolute path (to the `lib\plugins` directory) instead of a relative path.

**To run a full crawl:**

1. Open a command prompt.
2. Navigate to the Web Crawler root directory.
   Note that you can run the startup script from an external directory if you have set an absolute path in the `plugin.folders` configuration property.
3. Run the `web-crawler.bat` (for Windows) or `web-crawler.sh` (for UNIX) script with at least the `-d` and `-s` flags. You can use the optional flags to customize the crawl, such as using the `-w` flag to specify the workspace directory. For example:

   ```
   .\bin\web-crawler -c conf\web\myconfig -d 2 -s mysites.lst
   ```

   If the crawl begins successfully, you see the `INFO` progress messages.

The crawl is finished when you see the `Crawler complete` message from the Web Crawler. The output file is written to the `output` subdirectory in the workspace directory.

Note that by default, the console receives all messages. You can create a crawl log by either redirecting the output to a log (such as `>crawl.log`) or specifying a file appender in the `log4j.properties` logging configuration file.

Below is an example of a full crawl using the default polite configuration. For ease of reading, the timestamps and module names are truncated. The complete output will include the following summaries:

- Crawl metrics information (the `Perf` sections)
- Crawl progress information organized by host and seed depth

The crawl summaries include such page information as how many pages were fetched, redirected, retried, gone (i.e., pages were not available because of 404 errors or other reasons), and filtered.

**Example of running a full crawl**

```
C:\Endeca\CAS\3.0.2>.\bin\web-crawler -c ..\workspace\conf\web-crawler\po¬
lite-crawl -d 0 -s http://www.endeca.com
INFO    2009-07-27 09:38:47,528 0       com.endeca.itl.web.Main [main]
Adding seed: http://www.endeca.com
INFO    2009-07-27 09:38:47,544 16      com.endeca.itl.web.Main [main]
```

```
Seed URLs: [http://www.endeca.com]
INFO   2009-07-27 09:38:49,606 2078   com.endeca.itl.web.db.CrawlDbFactory
 [main]  Initialized crawldb: com.endeca.itl.web.db.BufferedDerbyCrawlDb
INFO    2009-07-27 09:38:49,606 2078    com.endeca.itl.web.Crawler
[main] Using executor settings: numThreads = 100, maxThreadsPerHost=1
INFO    2009-07-27 09:38:50,841 3313    com.endeca.itl.web.Crawler
[main] Fetching seed URLs.
INFO    2009-07-27 09:38:51,622 4094    com.endeca.itl.web.Crawler
[main] Seeds complete.
INFO    2009-07-27 09:38:51,653 4125    com.endeca.itl.web.Crawler
[main] Starting crawler shut down, waiting for running threads to complete
INFO    2009-07-27 09:38:51,653 4125    com.endeca.itl.web.Crawler
[main] Progress: Level: Cumulative crawl summary (level)
INFO    2009-07-27 09:38:51,653 4125    com.endeca.itl.web.Crawler
[main] host-summary: www.endeca.com to depth 1
host    depth   completed       total   blocks
www.endeca.com  0       1       1       1
www.endeca.com  1       0       38      1
www.endeca.com  all     1       39      2

INFO    2009-07-27 09:38:51,653 4125    com.endeca.itl.web.Crawler
[main] host-summary: total crawled: 1 completed. 39 total.
INFO    2009-07-27 09:38:51,653 4125    com.endeca.itl.web.Crawler
[main] Shutting down CrawlDb
INFO    2009-07-27 09:38:51,700 4172    com.endeca.itl.web.Crawler
[main] Progress: Host: Cumulative crawl summary (host)
INFO    2009-07-27 09:38:51,715 4187    com.endeca.itl.web.Crawler
[main]
Host: www.endeca.com:  1 fetched. 0.0 mB. 1 records. 0 redirected. 0 retried.
 0
gone. 19 filtered.
INFO    2009-07-27 09:38:51,715 4187    com.endeca.itl.web.Crawler
[main] Progress: Perf: All (cumulative) 2.0s. 0.5 Pages/s. 4.8 kB/s. 1
fetched. 0.0 mB.
 1 records. 0 redirected. 0 retried. 0 gone. 19 filtered.
INFO    2009-07-27 09:38:51,715 4187    com.endeca.itl.web.Crawler
[main] Crawl complete.
```

## Running resumable crawls

You run a resumable crawl from the command line.

You can run a resumable crawl if you use the same workspace directory as the previous crawl and if a valid history database exists in the `state/web` directory. The resumed crawl work runs any URL in the database that has a status of pending and also generates new URLs to crawl.

Keep in mind that the value of the `-d` flag should be greater than that of the previous crawl, or else no new records are retrieved (unless the previous crawl did not finish the depth). Also, you cannot change the seed. You can, however, change the configuration of the resumed crawl.

**Note:** If you are using the default configuration, Web crawls must be run from the Web Crawler root directory (i.e., in a Windows installation `\CAS\`*`version`* directory). To run crawls from other directories, you must change the `plugin.folders` configuration property so that it uses an absolute path (to the `lib\plugins` directory) instead of a relative path.

**To run a resumable crawl:**

1. Open a command prompt.
2. Navigate to the Web Crawler root directory.

   For example, in a default installation on Windows, this is `\CAS\version`.

   Note that you can run the startup script from an external directory if you have set an absolute path in the `plugin.folders` configuration property.
3. Run the `web-crawler.bat` (for Windows) or `web-crawler.sh` (for UNIX) script with the `-r` and `-d`) flags. Use the `-w` flag if you need to specify the location of the workspace directory. For example:

   ```
   .\bin\web-crawler -r -d 3
   ```

   If the crawl begins successfully, the first `INFO` message reads:

   ```
   Resuming an old crawl. Seed URLs are ignored.
   ```

The crawl is finished when you see the `Crawler complete` message from the Web Crawler. The output file is written to the `output` subdirectory in the workspace directory, while the previous output file is renamed and moved to the `output\archive` subdirectory.

Below is an example of a resumed crawl using the default polite configuration. For ease of reading, the timestamps and module names are truncated. As with full crawls, the complete output will include the crawl metrics and crawl host progress summaries.

**Example of running a resumed crawl**

```
C:\Endeca\3.0.2\CAS>.\bin\web-crawler -d 1 -c ..\workspace\conf\web-
crawler\polite-crawl -r
Resuming an old crawl.  Seed URLs are ignored.
Initialized crawldb: com.endeca.itl.web.db.BufferedDerbyCrawlDb
Using executor settings: numThreads = 100, maxThreadsPerHost=1
Resuming the crawl.
Starting crawler shut down, waiting for running threads to complete
Finished level: host: endeca.com, depth: 1, max depth reached
Progress: Level: Cumulative crawl summary (level)
host-summary: endeca.com to depth 2
host          depth   completed   total   blocks
endeca.com    0       0           0       0
endeca.com    1       36          36      1
endeca.com    2       0           141     1
endeca.com    all     36          177     2

host-summary: total crawled: 36 completed. 177 total.
Shutting down CrawlDb
Progress: Host: Cumulative crawl summary (host)
Host: endeca.com:  35 fetched. 0.4 mB. 35 records.
  0 redirected. 0 retried. 1 gone. 377 filtered.
Progress: Perf: All (cumulative) 40.0s. 0.9 Pages/s.
  9.6 kB/s. 35 fetched. 0.4 mds. 0 redirected.
  0 retried. 1 gone. 377 filtered.
Crawl complete.
```

# Record properties generated by a crawl

During a crawl, the Endeca Web Crawler produces record properties according to a standardized naming scheme.

The Web Crawler generates record properties and assigns them a qualified name with a period (.) to separate qualifier terms. The qualified name is constructed as follows:

- The first term is always `Endeca` and is followed by one or more additional terms.
- The second term describes a property category (for example, `Web` or `Document`).
- If present, the third and fourth terms fully qualify the property (for example, `Endeca.Web.URL.Pro¬tocol`).

Any of these properties can be mapped to Endeca properties or dimensions by the Property Mapper in your pipeline.

### Source-file properties

The following record properties describe the source of files that are fetched from a Web crawl.

| Endeca Property Name | Property Value |
|---|---|
| `Endeca.SourceType` | Indicates the source type of the crawl. The Web Crawler produces values with `Web`. |
| `Endeca.Id` | Provides a unique identifier for a record. `Endeca.Id` has the same value as `Endeca.Web.URL`. |
| `Endeca.Web.Accept-Ranges` | The value of the Accept-Ranges header field, which allows the server to indicate its acceptance of range requests for a resource. |
| `Endeca.Web.Connection` | The value of the Connection general-header field as returned from the server. |
| `Endeca.Web.Content-Type` | The value of the Content-Type header field, which indicates the media type of the entity-body. Examples of media types are `text/html` and `image/gif`. |
| `Endeca.Web.ETag` | The value of the ETag header field, which provides the current value of the entity tag for the requested variant. |
| `Endeca.Web.Host` | The Internet host and port number where the document resides. The absence of port information implies the default port for the service requested. |
| `Endeca.Web.HTTP.Content-Length` | The value of the Content-Length header field, which indicates the size of the entity-body. |
| `Endeca.Web.HTTP.Status` | The HTTP response status code, which determines the outcome of the request (for example, `200` indicates a successful request). |
| `Endeca.Web.Last-Modified` | The value of the Last-Modified entity-header field, which indicates the date and time at which the origin server believes the file was last modified. Typically, the value is the file system last-modified time. |
| `Endeca.Web.HTMLMetaTag.`*name* | The value of an HTML meta tag, where *name* is the name of the meta tag. For example, `Endeca.Web.HTML¬MetaTag.keywords` would contain the keywords defined in the tag. |
| `Endeca.Web.SeedUrl` | The URL of the seed from which this URL came. |

| Endeca Property Name | Property Value |
|---|---|
| `Endeca.Web.LinkedFromUrl` | The URL of the page that contained the outlink to this page. |
| `Endeca.Web.LinkedFromUrl.LinkText` | The text that was used on the `LinkedFromUrl` to link to this page. |
| `Endeca.Web.Server` | The value of the Server response-header field, which contains information about the software used by the origin server to handle the request (for example, `Apache-Coyote/1.1`). |
| `Endeca.Web.URL` | The URL of the document. |
| `Endeca.Web.URL.Protocol` | The protocol of the source document (for example, `http` or `https`). |

**Content properties**

The content properties contain information (including the text) of the document. Note that some of the properties are generated by the CAS Document Conversion Module.

| Endeca Property Name | Property Value |
|---|---|
| `Endeca.Document.CharEncodingForConversion` | The encoding used for text conversion of the document. |
| `Endeca.Document.Metadata.`*`attribute`* | Metadata information in the document. The metadata attributes depend on which ones were added by the authoring tool used to create the document. For example, an Adobe Acrobat PDF document could have such metadata attributes as `Endeca.Document.Metadata.title` and `Endeca.Document.Metadata.primary_author`. |
| `Endeca.Document.MimeType` | The MIME Type of the document, if it can be determined. Common examples of this property value include `text/html`, `application/pdf`, and `image/gif`. |
| `Endeca.Document.OriginalCharEncoding` | The original encoding of the body of the document, if it can be determined. This property value could be an ISO code or other encoding representation (for example, UTF-8, CP1252, or ISO-8859-1). |
| `Endeca.Document.Outlink` | A hypertext link (as an absolute URL) that references another document or another site. |
| `Endeca.Document.OutlinkCount` | The number of links (`Endeca.Document.Outlink` properties) in this document. |
| `Endeca.Document.Text` | The text (content) of the source document. Note that the Document Conversion Module typically does not preserve line break information. |
| `Endeca.Document.TextExtraction.Error` | An error that occurred during the parsing process, including errors returned by the Document Conversion Module. |
| `Endeca.Document.Title` | The title of the document. |

| Endeca Property Name | Property Value |
|---|---|
| `Endeca.Document.XHTML` | The content of the document in XHTML. This property is created only when the `output.dom.include` property is set to `true`. If it is, the Web Crawler normalizes the content of HTML documents to XHTML and stores it in this property. |
| `Endeca.File.Size` | The size of the file, as indicated by the size of the byte stream. |

### Character encoding maps

For the two encoding properties, the `OriginalCharEncoding` is retrieved from the content-type set in the HTTP header; if that fails, the Web Crawler tries to retrieve it from the downloaded content bytes.

The Web Crawler also keeps an alias map that maps character encodings which are often used in mislabelled documents to their correct encodings. The map is:

- ISO-8859-1 maps to windows-1252
- EUC-KR maps to x-windows-949
- x-EUC-CN maps to GB18030
- GBK maps to GB18030

If the encoding is mapped to a value, then `CharEncodingForConversion` is set to the mapped value; otherwise, it is set to the same value as the `OriginalCharEncoding` value.

Chapter 5

# Running the Sample Web Crawler Plug-in

This section provides instructions for running the sample Web Crawler plug-in, a custom parse filter plug-in that adds HTML meta tags as additional properties to the output records.

## About the Web Crawler plug-in framework

The Endeca Web Crawler is based on the Apache Nutch open-source project. As a result, its major functionality is implemented as plug-ins. Its framework allows you to write your own plug-ins, such as plug-ins that extract additional content from Web pages.

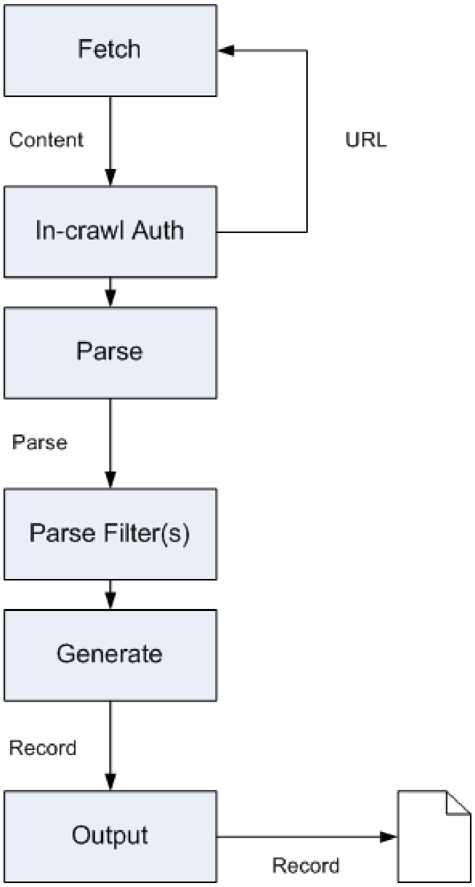
The sample plug-in demonstrates how to integrate custom plug-ins into the Web Crawler. The Endeca Web Crawler APIs contain sample code and documentation to help you create your own plug-ins.

All plug-ins (including the default plug-ins and user-created plug-ins) reside in the `CAS/`*`version`*`/lib/web-crawler/plugins` directory. Each individual plug-in directory contains one or more JAR files and a plug-in descriptor file (named `plugin.xml`).

### How the Web Crawler processes URLs

Knowing how the Web Crawler processes URLs helps you understand where a new plug-in fits in, because the URL processing is accomplished by a series of plug-ins.

Each URL is processed by a thread in the following manner:



The processing flow is as follows:

1. The scheduler determines which URL should be fetched (this step is not shown in the diagram).
2. Fetch: A `Protocol` plug-in (such as the `protocol-httpclient` plug-in) fetches the bytes for a URL and places them in a Content object.
3. In-crawl Auth: An `Authenticator` plug-in can determine whether form-based authentication is required. If so, a specific login URL can be fetched, as shown in the diagram.
4. Parse: A `Parse` plug-in parses the content (the Content object ) and generates a `Parse` object. It also extracts outlinks. For example, the `parse-html` plug-in uses the Neko library to extract the DOM representation of a HTML page.
5. Filter: `ParseFilter` plug-ins do additional processing on raw and parsed content, because these plug-ins have access to both the Content and Parse objects from a particular page. For example, the `endeca-xpath-filter` plug-in (if activated) uses XPath expressions to prune documents.
6. Generate: A record is generated and written to the record output file. In addition, any outlinks are queued by the scheduler to be fetched.

In the processing flow, the sample `htmlmetatags` plug-in would be part of step 5, because it does additional processing of the parsed content.

# About the sample custom filter plug-in

Custom filters (`ParseFilter`) implement content extensions. These filters can examine the contents of a page (either the raw page contents or the parsed DOM) and add additional properties to records that are produced.

These properties can augment records with additional information beyond generic HTML document properties (such as content size, encoding, and title).

The HTMLMetatagFilter plug-in illustrates how to add a custom plug-in to the Web Crawler. It is located in `CAS\version\sample\custom-web-crawler-plugin`.

By default, the Web Crawler does not return HTTP meta tags in the record output. The sample plug-in extends the default parsing of HTML documents by adding property values (i.e., HTML meta tags) to the Endeca records that the Web Crawler generates. These HTML meta tags include such data as keywords, descriptions, authors, and so on.

For example, an HTML page can have these meta tags:

```
<html>
<head>
<title>XYZ: The Worldwide Leader In Sports</title>
<meta name="description" CONTENT="XYZ.com provides sports coverage." />
<meta name="keywords" CONTENT="XYZ.com, sports scores, sports news" />
<meta name="robots" content="index, follow" />
<meta name="googlebot" content="index, follow" />
</head>
<body>
...
</body>
</html>
```

The HTMLMetatagFilter plug-in can add the properties to the `Parse` class's metadata object. These metadata properties are added to the Endeca record. For example:

```
...
<PROP NAME="Endeca.Document.HTML.MetaTag.description">
<PVAL>XYX.com provides sports coverage.</PVAL>
</PROP>
<PROP NAME="Endeca.Document.HTML.MetaTag.keywords">
<PVAL>XYX.com, sports scores, sports news.</PVAL>
</PROP>
...
```

# Adding a custom plug-in to the Endeca Web Crawler

This topic offers an overview of how to add your custom plug-in to the Endeca Web Crawler.

To add a custom plug-in to the Endeca Web Crawler:

1. Open Eclipse and load your custom plug-in project.
2. Use Eclipse to write the Java code for your new class.
3. Create the `plugin.xml` file for the new plug-in.
4. Build the JAR file for your new plug-in.
5. Create a directory for the plug-in (containing the JAR file and `plugin.xml`) and copy it to the `CAS/`*`version`*`/lib/web-crawler/plugins` directory.

6. Activate the plug-in by adding the plug-in ID to the `plugins.include` property in the `site.xml` configuration file.
7. Run the Web Crawler and verify that record output contains the new properties that the plug-in added.

See following topics for more detailed explanations of the above steps.

# Opening the sample plug-in project

For the purpose of this sample, you load the sample parse filter plug-in project. If you were creating your own plug-in, you would create your own Eclipse project.

To open the sample plug-in project:

1. Start Eclipse.
2. Import the sample plug-in project from the `custom-web-crawler-plugin` directory:
   a) Open the **File** menu.
   b) Click **Import**.
   c) Expand the **General** folder.
   d) Select **Existing Projects into Workspace** and click **Next**.
   e) Click **Browse** and navigate to `CAS\`*`version`*`\sample\custom-web-crawler-plugin`.
   f) Click **Ok**.
   g) Check **Copy projects into workspace** then click **Finish**.

Under the `filter-htmlmetatags` project, note the `src\java` directory that contains the `com.endeca.itl.web.parse` package and the source file.

# Overview of the sample HTMLMetatagFilter plug-in

For the purpose of this sample, we use the source for the `HTMLMetatagFilter` class that is in the `HTMLMetatagFilter.java` source file (in the `CAS/`*`version`*`/sample/custom-web-crawler-plugin/src` directory). If you were writing your own plug-in, you would write the code for your custom plug-in.

This source file can be used as a template for your custom plug-in package.

```
package com.endeca.itl.web.parse;

import java.util.Map;
import java.util.Properties;

import org.apache.hadoop.conf.Configuration;
import org.apache.nutch.parse.HTMLMetaTags;
import org.apache.nutch.parse.Parse;
import org.apache.nutch.parse.ParseData;
import org.apache.nutch.parse.ParseFilter;
import org.apache.nutch.protocol.Content;

public class HTMLMetatagFilter implements ParseFilter {

    public static String METATAG_PROPERTY_NAME_PREFIX = "Endeca.Docu¬
ment.HTML.MetaTag.";
```

```
    public Parse filter(Content content, Parse parse) throws Exception {
       parse.getData().getParseMeta().add("FILTER-HTMLMETATAG", "ACTIVE");

        ParseData parseData = parse.getData();
        if (parseData == null) return parse;

        HTMLMetaTags tags = parse.getData().getMetaTag();
        if (tags == null) return parse;

        Properties tagProperties = tags.getGeneralTags();
        for (Map.Entry<Object,Object> entry : tagProperties.entrySet()) {
           parse.getData().getParseMeta().add(METATAG_PROPERTY_NAME_PREFIX

                + (String)entry.getKey(), (String)entry.getValue());
            }
            return parse;
        }

        public Configuration getConf() {
            return null;
        }

        public void setConf(Configuration conf) {
        }
}
```

The code works as follows:

1. The `Metadata.add()` method adds a metadata name/value mapping to the `Parse` object (the name is `FILTER-HTMLMETATAG` and value is `ACTIVE`). You can leave in this line in the code when you first run the custom parse-filter plug-in, to verify that the objects are being updated. After you are satisfied that the plug-in is running correctly, you can remove the line from the code.

   ```
   parse.getData().getParseMeta().add("FILTER-HTMLMETATAG",
   "ACTIVE");
   ```

2. The `Parse.getData()` method returns a `ParseData` object, which contains data extracted from a page's content. Because plug-ins should be programmed in a defensive manner, the object is checked to make sure that it is not null before proceeding.

   ```
   ParseData parseData = parse.getData();
   if (parseData == null) return parse;
   ```

3. The `ParseData.getMetaTag()` method returns an `HTMLMetaTags` object, which holds the information about HTML meta tags extracted from a page. Note that this method has been added by Endeca, and is therefore not part of the original Nutch API. The object is then checked to ensure that it contains data.

   ```
   HTMLMetaTags tags = parse.getData().getMetaTag();
   if (tags == null) return parse;
   ```

4. The `HTMLMetaTags.getGeneralTags()` method returns a `Properties` object containing all of the properties.

   ```
   Properties tagProperties = tags.getGeneralTags();
   ```

5. Iterate through the properties. For each name/value pair, add a new entry to the Parse's Metadata object (accessed via the `ParseData.getParseMeta()` method). The `MetaData.add()` method actually adds the metadata name/value mapping.

   ```
   for (Map.Entry<Object,Object> entry : tagProperties.entrySet()) {
   parse.getData().getParseMeta().add(METATAG_PROPERTY_NAME_PREFIX
   + (String)entry.getKey(), (String)entry.getValue());
   }
   ```

When the class finishes, it returns the modified `Parse` object.

# Overview of the plugin.xml file

The `plugin.xml` file describes the plug-in to the Web Crawler. The file resides in the plug-in directory along with the JAR file.

The following is the `plugin.xml` file that is included with the `HTMLMetatagFilter` project:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
id="filter-htmlmetatags"
name=""
version="1.0"
provider-name="com.endeca.itl.web">
<runtime>
<library name="filter-htmlmetatags.jar">
<export name="*"/>
</library>
</runtime>
<requires>
<import plugin="nutch-extensionpoints"/>
</requires>
<extension id="com.endeca.itl.web.parse.HTMLMetatagFilter"
name="HTML Metatag filter"
point="org.apache.nutch.parse.ParseFilter">
<implementation id="filter-htmlmetatags"
class="com.endeca.itl.web.parse.HTMLMetatagFilter">
</implementation>
</extension>
</plugin>
```

The file defines the name of the JAR (`filter-htmlmetatags.jar`), the name of the extension point (`ParseFilter`), and the name of the implementing class (`HTMLMetatagFilter`). It also sets the ID of the plug-in (with the `<plugin id>` attribute); you set this ID in the configuration file, as shown later.

# Building the sample plug-in

For the purpose of this sample, use Eclipse to build a JAR of the sample Web Crawler parse plug-in.

To build the sample plug-in JAR file in Eclipse:

1. Right-click on the source file and select **Export**.
2. Select **Java** > **JAR file** and click **Next**.

3. In the **JAR File Specification** dialog:
    a) Select the resources to export (the com.endeca.itl.web.parse package).
    b) If it is not already checked, select **Export generated class files and resources**.
    c) Select an export destination.
    d) Select **Compress the contents of the JAR file**.
    e) Click **Next** when you are satisfied with the specification.
4. Select any other JAR Packaging Options that you want and click **Finish**.

The next step is to copy the files to the proper locations in the CAS directory.

# Adding the plug-in to the CAS lib directory

After you build the Jar for your custom plug-in, create a directory for the plug-in and copy this to the Web Crawler's plug-in directory.

To add your plug-in to the CAS `lib` directory:

1. Create a directory of the same name that is specified in the `plugin id` element in the `plugin.xml` file. For example, the sample plug-in uses `filter-htmlmetatags` as the name of the directory.
2. Copy the Jar and the `plugin.xml` file into the directory.
3. Copy the directory to the `CAS\`*`version`*`\lib\web-crawler\plugins` directory.

The next step is to activate the plug-in for the Web Crawler.

# Activating the plug-in for the Web Crawler

Oracle recommends that you modify the crawl-specific `site.xml` file, rather than the global `default.xml` file (this is because the `site.xml` settings override the `default.xml` global settings).

Use the following steps to activate the plug-in.

To activate the plug-in for the Web Crawler:

1. From `default.xml` (located in `CAS\workspace\conf\web-crawler\default`), copy the `plugin.includes` and `plugin.excludes` properties to `site.xml` (located in `CAS\workspace\conf\web-crawler\polite-crawl` or `CAS\workspace\conf\web-crawler\non-polite-crawl`).
2. Add the plug-in ID to the `plugin.includes` property in the `site.xml` file, as shown in this abbreviated example:

```
...
<property>
  <name>plugin.includes</name>
  <value>filter-htmlmetatags|... | output-endeca-record</value>
  <description>
    Regular expression naming plugin directory names to include.
  </description>
```

```
</property>
...
```

**Note:** The value name (`filter-htmlmetatags` in this example) must refer to the plug-in ID as set in the plug-in's `plugin.xml` definition file.

3. Check both configuration files (`default.xml` and `site.xml`) for the `plugin.excludes` property and make certain that the plug-in ID is not excluded, as in the following example:

```
...
<property>
  <name>plugin.excludes</name>
  <value></value>
  <description>
    Regular expression naming plugin directory names to exclude.
  </description>
</property>
```

4. Check the parse filtering order. If you are using the `parser.filters.order` configuration property to specify the order by which parse filters are applied, make sure that you include the `filter-htmlmetatags` in the property value. If you are not using this property (i.e., it has an empty value), you can leave the property as-is.

You can now run the Web Crawler with the new plug-in.

# Running the Web Crawler with the new plug-in

After you activate the new plug-in, you can run new crawls exactly as before.

1. Run the Web Crawler.
2. Examine the record output to verify that returned records contain the new properties from the plug-in.

**Example of a record returned with new properties**

In this example, the “description” and “keywords” meta tags are returned. A returned record with the new `Endeca.Document.HTML.MetaTag` properties looks as follows:

```
...
<PROP NAME="Endeca.Document.HTML.MetaTag.description">
  <PVAL>XYX.com provides sports coverage.</PVAL>
</PROP>
<PROP NAME="Endeca.Document.HTML.MetaTag.keywords">
  <PVAL>XYX.com, sports scores, sports news.</PVAL>
</PROP>
...
```

# Index