Oracle® Endeca Information Discovery Integrator

Integrator Acquisition System Web Crawler Guide

Version 3.0.0 • May 2013

ORACLE

Copyright and disclaimer

Table of Contents

Preface

Oracle® Endeca Information Discovery Integrator provides a suite of products to load data from disparate source systems and store it for use in an Endeca Server data domain. The Integrator products include:

- Integrator ETL - Integrator ETL is a high-performance data integration platform that lets you extract source records from a variety of sources and sends that data to the Data Ingest Web Service, which in turn loads the records into the Oracle Endeca Server.
- Integrator Acquisition System - The Integrator Acquisition System, or IAS, is a set of components that crawl source data stored in a variety of formats including: file systems, delimited files, JDBC databases, and custom data sources. IAS transforms the data, if necessary, and outputs the data to an XML file or a Record Store instance that can be accessed by Integrator ETL for use in the Endeca Server.
- IKM SQL to Endeca Server - provides integration and loading modules that enable writing source data to an Endeca Server target within Oracle Data Integrator.

About this guide

This guide describes how to configure the Endeca Web Crawler and run it to gather source data from Web sites.

It assumes that you are familiar with the concepts of the Endeca Integrator Acquisition System and the Endeca Information Discovery Integrator.

Who should use this guide

This guide is intended for data developers who are responsible for running Web crawls and making the resulting data available for use in Endeca Information Discovery Integrator.

Conventions used in this guide

The following conventions are used in this document.

Typographic conventions

The following table describes the typographic conventions used in this document.

Table 0.1: Typographic conventions

Typeface	Meaning
User Interface Elements	This formatting is used for graphical user interface elements such as pages, dialog boxes, buttons, and fields.
`Code Sample`	This formatting is used for sample code phrases within a paragraph.
`<Variable Name>`	This formatting is used for variable values, such as <install path>.

Typeface	Meaning
`File Path`	This formatting is used for file names and paths.

Symbol conventions

The following table describes symbol conventions used in this document.

Table 0.2: Symbol conventions

Symbol	Description	Example	Meaning
>	The right angle bracket, or greater-than sign, indicates menu item selections in a graphic user interface.	File > New > Project	From the File menu, choose New, then from the New submenu, choose Project.

Contacting Oracle Customer Support

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

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

Chapter 1

Introduction

This section provides introductory information about the Endeca Web Crawler.

Web Crawler overview

The Endeca Web Crawler is installed by default as part of the IAS installation. The Web Crawler gathers source data by crawling HTTP and HTTPS Web sites and writes the data in a format that is accessible to Endeca Information Discovery Integrator (either XML or a Record Store instance).

After the Web Crawler writes the Endeca records, you can configure an Endeca Record Store Reader component (in Integrator) to read the records from a Record Store instance into an Integrator graph. This is the recommended integration model.

Although you can process XML records in an Integrator graph, this model requires more configuration to create XML mappings using the XMLExtract component. XML output is typically used as a convenient format to examine the records after a Web crawl.

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

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.

Plug-in Support

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 plug-ins. Plug-ins provide a means to extract additional content, such as HTML meta tags, from Web pages.

SSL Support

You can configure the Endeca Web Crawler to read and write from an SSL-enabled Record Store instance. For details, see the "Configuring SSL in the Integrator Acquisition System" chapter of the *Security Guide for Integrator*.

Running a sample Web crawl of endeca.com

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

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

The sample crawl is configured to output the records as uncompressed XML. The XML format allows you to easily read the output file 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 window.
2. Change to the `<install path>\IAS\<version>\bin` directory.
3. Run the `web-crawler` script with the `-d` flag set to 0 to crawl only the root of the site.
   Here is a Windows example:

   ```
   web-crawler -c C:\Oracle\Endeca\IAS\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 `<install path>\IAS\<version>\bin\polite-crawl-workspace\output` directory.

Running a sample Web crawl that writes to a Record Store

The sample Web crawl runs the Endeca Web Crawler and writes output to a Record Store instance instead of to a file on disk. This sample is stored in `IAS\<version>\sample\webcrawler-to-recordstore`. The `run-sample` script runs the sample Web Crawler.

The directory also contains a `recordstore-configuration.xml` file that is configured for records produced by the Web Crawler. In particular, the file has these two Record Store configuration properties:

```
<changePropertyNames/>
<idPropertyName>Endeca.Id</idPropertyName>
```

Setting the `idPropertyName` is important because the Record Store instance generates a unique record ID based on the property value.

The sample Web crawler writes output directly to a Record Store instance. The `site.xml` file, in the `IAS\<version>\sample\webcrawler-to-recordstore\conf` directory, has three output properties that specify the Record Store information:

```
<property>
       <name>output.recordStore.host</name>
       <value>localhost</value>
       <description>
       The host of the record store service.
       Default: localhost
       </description>
</property>

<property>
       <name>output.recordStore.port</name>
       <value>8510</value>
       <description>
```

```
       The port of the record store service.
       Default: 8510
       </description>
</property>

<property>
       <name>output.recordStore.instanceName</name>
       <value>rs-web</value>
       <description>
       The name of the record store service.
       Default: rs-web
       </description>
</property>
```

Be sure to change the values if you create a Record Store instance on a different host name and port.

To run the sample Web crawl:

1. Start the Endeca IAS Service if it is not already running.
   - Windows: Start the IAS Service from the Windows Services console.
   - UNIX: Run the `ias-service.sh` script.
2. Open a command prompt window.
3. Change to the `IAS\<version>\sample\webcrawler-to-recordstore` directory.
4. Run the `run-sample` script.

When the Web Crawler finishes, the output is written to the Record Store, instead of to a file on disk. If you check `ias-service.log`, you should see these messages similar to this example:

```
Starting new transaction with generation Id 1
Started transaction 1 of type READ_WRITE
Marking generation committed: 1
Committed transaction 1
```

In the example, the Record Store is storing the record generation with an ID of 1.

Chapter 2

Configuration

This section provides reference information to configure 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 contains default configuration properties for all of your crawls. 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.
`site.xml`	A per-crawl configuration file. The settings in this file override those in the `default.xml` file. This file is used to adjust per-crawl settings.
`crawl-urlfilter.txt`	Contains a list of regular expressions that include and exclude 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`.

Configuration Filename	Purpose
`regex-normalize.xml`	Contains a list of URL normalizations, which 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 that IAS can recognize. 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 IAS, 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 no fetcher delay (versus a 1-second delay in the polite version) and a maximum of 52 threads (versus 1 in the polite version).

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 the 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.

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.

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.

Configuration files

The site.xml file

The crawl-urlfilter.txt file

The regex-normalize.xml file

The mime-types.xml file

The parse-plugins.xml file

The form-credentials.xml file

The log4j.properties file

Enabling the IAS Document Conversion Module

Disabling the IAS Document Conversion Module

About document conversion options

Configuring Web crawls to write output to a Record Store instance

HTTP Properties

Authentication properties

Properties for authenticated proxy support

Fetcher properties

URL normalization properties

MIME type properties

Plugin properties

Parser properties

Parser filter properties

URL filter properties

Crawl scoping properties

HTTP Properties

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

Property Name	Property Value
`http.agent.name`	Required. 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.
`http.robots.ignore`	Specifies whether the crawler ignores `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`	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. This is a Boolean value with a default of `true`.
`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.

Property Name	Property Value
`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. Oracle 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.
`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.

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 a crawl downloads 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, 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 crawling content that exceeds 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.eidi.web.UrlProcessor
Content limit exceeded for http://xyz.com/pdf/B2B_info.pdf. Page is skipped.
```

Authentication properties

You 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.

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. In the `http.auth.form.credentials.file` property, 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 configure authenticated proxy support in the `default.xml` file.

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.

Property Name	Property Value
`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 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 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 `RegexUrlNormalizer` 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 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.

MIME type properties

You 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 and close the file.

Plugin properties

You 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 `IAS\<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 specifies 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 absolute 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 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.
`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.  **Note:** This may lead to undesirable behavior, such as submitting empty forms during the next fetch cycle. 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.  **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 specifies 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.eidi.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, 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 configure how the URL filter plugins are handled in the `default.xml` file.

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.sample.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 apply URL filters to the seeds with the `urlfilter.filter-seeds` property.

By default, the seeds are read in as-is (assuming 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.

Crawl scoping properties

You implement crawl scoping to control which URLs are crawled in the `default.xml` file..

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.

Property Name	Property Value
`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.

You configure a crawl's scope using 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 the following TLD names in the `default.xml` configuration file:

- 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 set the document conversion properties in the `default.xml` file.

The Endeca Web Crawler uses the IAS Document Conversion Module to perform text extraction on any document that is not: HTML, SGML, XML, text, or JavaScript. By using the properties listed in the table, you can configure the behavior of this module.

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.

Note that the IAS 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 IAS 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.

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, any content longer than the setting is not downloaded and you will see a WARN message similar to this example:

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

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

Output properties

You 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.

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 specifies a list of record properties that you want to exclude 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 `default.xml`.

The `default.xml` file 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.

Configuration files

The default.xml file

The site.xml 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 set the `crawl-urlfilter.txt` files to limit a crawl to a specific domain.

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 specify hosts to accept:

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 specifying regular expressions, make sure that you list 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 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 IAS 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 *Integrator Acquisition System 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 (&).

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.

The form-credentials.xml file

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 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.eidi.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.
<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.

Element	Meaning
<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 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

You 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.eidi.web.metrics=INFO

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%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.eidi.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.ConversionPattern=%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 IAS Document Conversion Module

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

Disabling the IAS Document Conversion Module

If desired, you can disable the IAS 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>\IAS\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 IAS 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>\IAS\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 IAS 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 IAS Document Conversion Module by modifying JVM property names and values.

Note that you cannot set these options in the standard configuration files.

The two options are:

- `stellent.fallbackFormat` determines the fallback format, that is, what extraction format will be used if the IAS 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. The default value is `none`.
- `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. The default value is `extended`.

Configuration files

The default.xml file

The site.xml file

The crawl-urlfilter.txt file

The regex-normalize.xml file

The mime-types.xml file

The parse-plugins.xml file

The form-credentials.xml file

The log4j.properties file

Enabling the IAS Document Conversion Module

Disabling the IAS Document Conversion Module

Configuring Web crawls to write output to a Record Store instance

Setting document conversion options

Setting document conversion options

Set the document conversion options as parameters to the JVM's `-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.

About document conversion options

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 describes how to modify 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:

1. You create and configure a Record Store instance to receive the Web Crawler output.
2. 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.

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

1. Start the Endeca IAS Service if it is not running already
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>\IAS\<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`:

   ```
   component-manager-cmd.bat create-component
   -h localhost -n WebCrawlerOutput -p 8510 -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.eidi.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>\IAS\<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`:

   ```
   recordstore-cmd.bat set-configuration
   -a WebCrawlerOutput -f C:\sample\webcrawler\recordstore-configuration.xml
   ```

   The command prompt displays:

   ```
   Successfully set recordstore configuration.
   ```