DRE 4 / AXE

version 4.3.x

Administration

Table of Contents

Autonomy

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

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

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

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

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

Europe and South Pacific

Autonomy Systems Ltd.

Cambridge Business Park

Cowley Road

Cambridge

CB4 0WZ

Help Desk:	+44 (0) 800 0 282 858	
Switchboard:	+44 (0) 1223 448 000	
Fax:	+44 (0) 1223 448 001	
E-mail	for information:	autonomy@autonomy.com
	for support:	uksupport@autonomy.com

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

Website: www.autonomy.com

USA

Autonomy Inc.

301 Howard Street

22nd Floor

San Francisco

CA 94105

Help Desk:	+1 877 333 7744	
Switchboard:	+1 415 243 9955	
Fax:	+1 415 243 9984	
E-mail	for information:	info@us.autonomy.com
	for support:	support@us.autonomy.com

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

Website: www.autonomy.com

Welcome

Thank you for choosing Autonomy and welcome to your DRE ™ version 4.3 Administrator's Guide.

Autonomy Solutions

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

Autonomy solutions include:

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

1. Introduction

At the heart of Autonomy’s software infrastructure is the DRE (Dynamic Reasoning Engine). The DRE is a scalable, multithreaded process which is based on advanced pattern-matching technology that exploits high-performance probabilistic modeling techniques.

The DRE performs the following core information operations:

Concept matching

The DRE accepts a piece of content (a sentence, paragraph or page of text, the body of an e-mail, a record containing human readable information, or the derived contextual information of an audio or speech snippet) or reference (identifier) as input and returns references to conceptually related documents ranked by relevance, or contextual distance. This is used to generate automatic hyperlinks between pieces of content.

Agent creation

The DRE provides the conceptual information that is needed to create agents.
The DRE accepts a piece of content (training text, a document or a set of documents) or reference (identifier) and returns an encoded representation of the concepts, including each concept’s specific underlying patterns of terms and associated probabilistic ratings.

Agent retraining

The DRE accepts an agent and a piece of content (training text, a document or a set of documents) and adapts the agent using the content.

Agent matching

The DRE accepts an agent and returns similar agents ranked by conceptual similarity. This is used to discover users with similar interests, or find experts in a field.

Agent alerting

The DRE accepts a piece of content (a sentence, paragraph or page of text, the body of an e-mail, a record containing human readable information, or the derived contextual information of an audio or speech snippet) and returns similar agents ranked by conceptual similarity. This is used to discover users who are interested in the content, or to find experts in a field.

Categorization

The DRE accepts a piece of content and returns categories ranked by conceptual similarity. This is used to find out which categories the content is most appropriate for, allowing subsequent tagging, routing or filing.

Summarization

The DRE accepts a piece of content and returns a summary of the information containing the most salient concepts of the content. In addition, summaries can be generated that relate to the context of the original inquiry - allowing the most applicable dynamic summary to be provided in the results of a given inquiry.

Clustering

The DRE, in conjunction with the Classification Server module, organizes large volumes of content or large numbers of profiles into self-consistent clusters. Clustering is an automatic agglomerative technique which partitions a corpus by grouping together information containing similar concepts.

Active matching

The DRE accepts textual information describing the current user task and returns a list of documents ordered by contextual relevance to the active task.

Retrieval

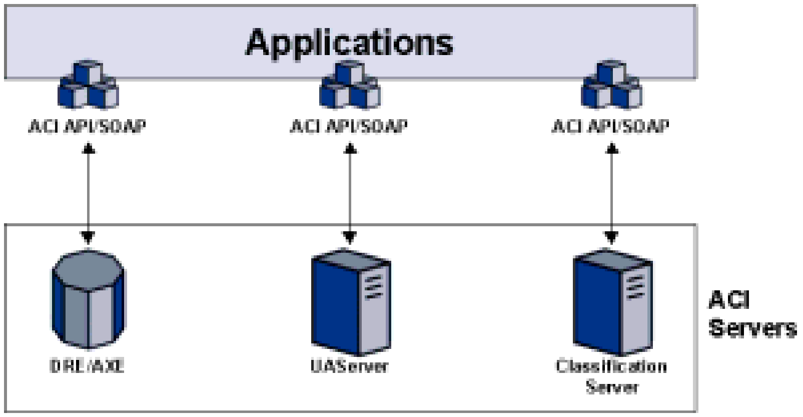
The DRE accepts a Boolean term or natural language query and returns a list of documents containing the terms ordered by contextual relevance to the query.

Automatic language detection

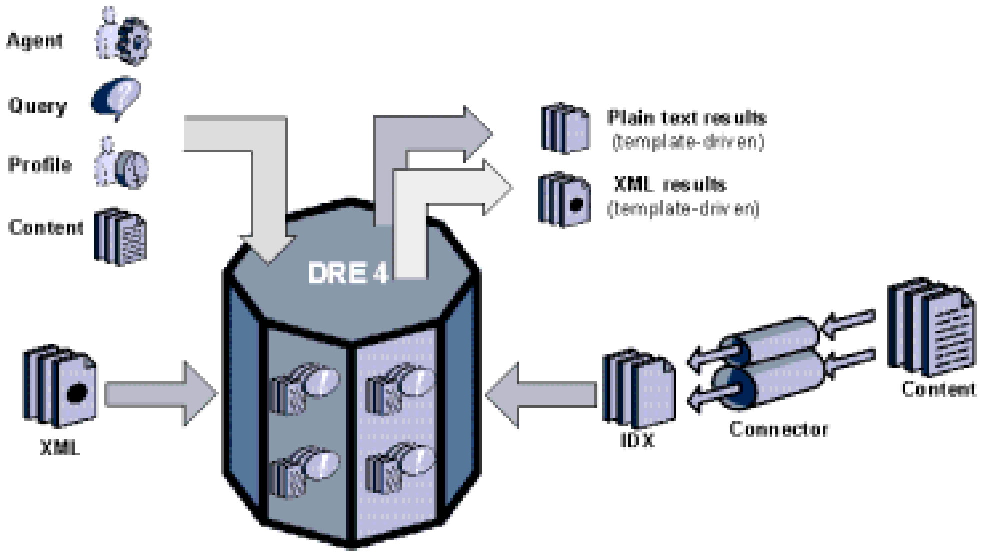
The DRE automatically identifies the language type (language and encoding) of files that are indexed and applies appropriate processes to them.

System architecture

The DRE uses the ACI (Autonomy Content Infrastructure) Client API to communicate with custom-built applications that retrieve data using HTTP commands. This communication is implemented over HTTP using XML and can adhere to SOAP:



Documents in IDX or XML format are indexed into the DRE (directly or using a Connector). The DRE stores the concepts of the document, and in response to queries, agents, profiles or content returns a link to the result document as well as a percentage weighting, which indicates how relevant this result document is to the original query. It can return results not only as plain text but also as XML (even if the documents was not in XML format when it was indexed):


Agent
Query
Profile
Content
Plain text results (template-driven)
XML results (template-driven)
DRE 4
XML
IDX
Connector
Content

Query types

Once the DRE has been configured for a particular installation, it operates automatically, handling a number of different query types:

Natural language
Natural language text is submitted as a query.

Boolean and bracketed Boolean
Standard Boolean AND/OR/NOT searches.

Fuzzy queries
If a search string is not quite accurate (for example, if it contains spelling mistakes) a fuzzy query returns results that contain words, which are similar to the entered string.

Proximity search
Words that appear close together in the search string are given a higher weighting.

Soundex keyword search
If the spelling of a keyword is not quite accurate but phonetically correct a Soundex keyword search returns results that contain the keyword and phonetically similar keywords (using a Soundex algorithm).

Proper names
Names are recognized and treated as a unit.

Synonym queries
A synonym query returns results which are conceptually similar to the query's terms and / or conceptually similar to the synonyms that are available for the query's terms.

Communicating with the DRE

You can communicate with the DRE using ACI (Autonomy Content Infrastructure) action commands.

To ensure backwards compatibility (if you are upgrading from a DRE 3), you can also use qmethods to operate the DRE. This allows you to operate the DRE 4 the same way you would a DRE 3. However, not all the DRE 4's functionality is available for qmethods. Also, in many cases the DRE will not respond to qmethods as quickly as to ACI action commands.

Depending on how you communicate with the DRE the following functionality is available.

Functionality available for ACI action and qmethod communication:

- Concept matching
- Agent creation
- Agent matching
- Agent retraining
- Agent alerting
- Categorization
- Summarization
- Clustering
- Active matching
- Retrieval
- Natural Language queries
- Boolean and bracketed Boolean queries

- Fuzzy queries
- Proximity search
- Soundex keyword search (expanded in DRE 4)
- Proper names queries
- Thesaurus / Synonym queries
- XML indexing
- Handling of multiple languages
- Fields printing
- Storage of all fields by default.
- Automatic language detection at index time

  The DRE can automatically detect the language and encoding of a buffer that you pass to it.

Functionality available for ACI action communication only:

- Automatic language detection at query time

  The DRE can automatically detect the language and encoding of a piece of text.
- Direct indexing of well formed XML with no import step
- XML output of information and results
- Scheduled expiration of individual documents

- Scheduled backups of the DRE

- Scheduled compression of the DRE

- Defined log streams

- Automatically detection of documents' built-in attributes and properties

- Compound sorting

  Result sorting can be based on more than one field at a time.

- Updated weighting algorithm

  Relevance is calculated using the results of latest research.

- Agent Boolean matching

  Boolean queries can be specified within a document, that must be matched by the query text before it can be returned.

- Inverted agent matching

  Queries can be weighted as if the document were the query text and the query text were in the DRE.

- Case sensitive matching

  Field text can be specified so that it only matches documents with exactly the same case.

- Combine on any reference field

  Results can have duplicates removed by any reference field.

- Highlighting

  Terms or sentences, within a document or buffer of your choice, can be highlighted if they satisfy certain criteria.

- Restriction by encoding / language

  Results can be restricted to a specific language, encoding or language type.

- Restriction by number of links

  Results can be restricted to documents that contain a certain number of link terms.

- Restriction by document ID

Results can be restricted to fall in a certain range of document ID.

- Minimum term length

  Query text terms that don't have a certain length can be ignored.

- Field printing

  The DRE can return specific fields, all fields or combine content to a single field.

- Sorting

  Results can be sorted by any field numerically or alphabetically, or by document ID, as well as the usual methods.

- Term information

  The DRE can retrieve the total number of occurrences, the APCM weighting, and document occurrences of any or all terms, sorted by any of the values.

- Term information in documents

  The DRE can retrieve the number of occurrences, the APCM weighting, and document occurrences of terms within only a subset of DRE documents.

- Result biasing

  Results can be biased on a sliding scale of the your choice, according to your value in a certain field.

- Bitwise field matching

  Decimal as well as hexadecimal bitwise field matching.

- Empty field detection

  Field specifiers exist to detect whether a document does or does not contain certain fields, and also to detect whether the field is empty or not.

- Exact term matching

  Documents can be matched according to the exact form of a term, prior to stemming.

- Relevance of field text

  Conceptual matching of the field text terms can be made to affect the overall relevance, or indeed not to affect it.

- Wildcard handling

  Wildcard terms can now contain characters in all encodings.

- Improved performance

  Performance with DRE4 is significantly faster than DRE3 for many cases, including:

  - Field text with many terms
  - Field text with MATCH or TERM specifiers
  - Requesting the total result count
  - Matching of numerical terms within fields

2. Installing DRE 4 / AXE

The DRE 4 / AXE should be installed by the system administrator.

System requirements

Platforms supported
Microsoft Windows NT 4, 2000 and XP
Linux
Solaris
AIX 4.3
HPUX 11
Tru64

Note: DRE 4 / AXE also supports other POSIX UNIX versions on request.

Minimum server specification
Pentium CPU
128 MB RAM
Intel or Intel Compatible Processor

Note: this specification is dependent on the amount of data that is stored in the DRE. If you want to run Autonomy connectors with your DRE, you should note that due to substantially different disk usage patterns it is beneficial to run connector and DRE processes on separate drives or partitions.

Installing the DRE on Windows

To install under Windows insert the DRE 4 CD-ROM into your CD-Rom Drive, and start the installation by double-clicking on the DRE4.3.x.EXE program in the root directory of the CD-ROM through Windows explorer.

Read and follow all installation instructions on the screen carefully.

1. The installation opens with the **Welcome** dialog. Read the text and click on **Next**.
2. The **License Information** dialog is displayed.

   Check the license details and click on **Next**. Contact Autonomy Technical Support if they are incorrect or if you are missing the licensekey.txt file that contains your license key.

   If you are installing the DRE 30 day evaluation version, enter the license holder.
3. The **License Agreement** dialog is displayed. Read the license agreement and click on **Next** to accept it.
4. The **Installation name** dialog is displayed.

   Enter a unique name for the DRE installation, and click on **Next**.

   Note: The unique name must not contain any spaces.
5. The **Choose Destination Location** dialog is displayed.

   Select the directory in which you want to install the DRE, and click on **Next**. By default the DRE is installed on **C:\Autonomy\myDRE**, but you can use the **Browse** button to navigate to another location.
6. The **DRE Details** dialog is displayed.

   Enter the following details and click on **Next**.

   Query Port Number
   The port number that is used to send queries to the DRE. By default this is **9000**.

   Note that this port is only used to support the backwards compatibility of qmethods.

   Index Port Number
   The port number that is used to index documents into the DRE (and to administer the DRE). By default this is **9001**.

   ACI Port Number
   The port that client machines use to send action commands to the DRE. By default this is **9002**.

   Database Name
   The name of the database that you want the DRE to contain. By default this is **Archive**.

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

    DiSH Service Port

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

    Service Control Clients

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

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

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

    Service Status Clients

    Enter the IP address of machines that are permitted to access the DRE's status but are not permitted to control it.

    Note: If you want to permit a number of machines to access the DRE's status, you can use a wildcard in the IP address.

    For example: Enter **187.*.*.*** to permit any machine whose IP address begins with 187 to access the DRE's status.

8. The **Dynamic Reasoning Engine Services** dialog is displayed.

    Specify whether you want to start the DRE service immediately after the installation is complete, and click on **Next**.

9. The **Extended Language Settings** dialog is displayed.

    If you are using one or more of the listed languages, check the appropriate boxes, and click on **Next**. This ensures that the extra files, which the listed languages require, are installed.

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

    Click on **Next** to confirm the settings you have made and start the installation. Alternatively click on **Back** to return to previous dialogs if you want to make any changes.

11. The **Installing** dialog is displayed.

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

12. The **Select Program Manager Group** dialog is displayed.

    Select the Program Manager group that you want to add DRE icons to.

13. The **Add Shortcuts** dialog is displayed.

    Specify whether you want to add DRE 4 shortcuts to the **Start** menu, and click on **Next**.

14. The **Installation Complete** dialog is displayed. DRE 4 has been installed successfully. Click on **Finish** to exit the installation.

Windows directory structure

Once the installation of DRE 4 is completed, your installation directory will contain the following files and subdirectories (note that **bold** font indicates folders):

examplecfgs	Folder that contains various example configuration files.
langfiles	Folder in which the data and text files are stored that the DRE requires to run multiple languages and convert encodings.
logs	Folder in which the DRE log files are stored.
templates	Folder in which templates are stored that are required to display results.
<InstallationName>.exe	DRE 4 executable.
<InstallationName>.cfg	Configuration file that contains the DRE 4 settings.
Copy of <InstallationName>.cfg	Safe copy of the configuration file that contains the DRE 4 settings.
<InstallationName>admin.exe	DRE Administration dialog executable.
<InstallationName>admin.chm	DRE Administration dialog online help.
importslave.exe	Import wizard executable (used by the DRE Administration dialog).
importwizard.exe	Import wizard executable (used by the DRE Administration dialog).
autonomy.css	Style sheet that the DRE4 / AXE HTML help uses.
various html files	DRE4 / AXE HTML help files.
<InstallationName>.lck	Internal DRE lock file.
<InstallationName>.str	Internal DRE structure file.
<InstallationName>.log	Initial DRE start up log file.
Install.log	Log file in which the indexing details are logged.
Uninstall.exe	Executable to uninstall DRE 4 from your computer.

Installing the DRE on UNIX

To install under Unix insert the DRE 4 CD-ROM into your CD-ROM Drive. Read and follow all installation instructions on the screen carefully.

1. Copy the DRE 4 installer from the CD to your local disk.
2. Uncompress the installer using the command:

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

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

   ```
   tar –xvf <Installer>.tar
   ```

   This creates a subdirectory called **AXE-4.3.x-<UNIX platform>**, which contains the following files:

   AXE.cfg

   AXE.exe

   LICENSE.TXT

   Setup.sh

   Start.sh

   Stop.sh

   Uninstall.sh

   edit_ini.exe

   examplecfgs

   freplace.exe

   And the subdirectories:

   examplecfgs

   multilingual

   stoplists

   templates

   unicodetables

4. Enter the command **cd AXE-4.3.x-<UNIX platform>** to move to this directory.
5. Run the installer script, **./Setup.sh**.

   The **Welcome** text is displayed. Read the text and then enter **v** to display the license agreement.

6. The license agreement is displayed. Read the agreement and enter **y** to accept it. Alternatively, enter **n** to cancel the installation.
7. Select the installation actions that you want to perform. Enter **1** to install the DRE or enter **2** to exit the installation.
8. If you are installing the DRE, enter the following installation details:

   The base name of the Autonomy XML Engine

   Enter a unique name for this installation of the DRE. By default this is **MyAXE**. **Note:** this name must not contain any spaces.

   The name of the directory in which you want to install the Autonomy XML Engine

   Enter the full path to the main installation directory for the DRE. By default this is **/Autonomy/<Base_Name>/**
9. Enter the following **AXE Settings**:

   The query port for the Autonomy XML Engine

   Enter the port number used to send queries to the DRE. By default this is **9000**.

   The index port for AXE

   Enter the port number that is used to index documents into the DRE (and to administer the DRE). By default this is **9001**.

   The ACI port for AXE

   Enter the port that client machines use to send action commands to the DRE. By default this is **9002**.

   The Service port for AXE

   Enter the port number that the DRE will use for DiSH communication. By default this is **10000**.
10. Enter **y** to specify that you want to install support for the following languages (Alternatively, enter **n** if you do not want to install support for a language):

    Chinese

    Japanese

    Korean

    Thai
11. The settings that you have entered are displayed. Check that your settings are correct, and press **Enter** to confirm your settings and to install the DRE. If you want to change a setting, enter the corresponding number, press **Enter** and then enter a **New value** for the setting. Alternatively, type **X** or press **Ctrl+C** to cancel the installation.
12. The installation has been completed successfully.

UNIX directory structure

Once the installation of DRE 4 is completed, your installation directory will contain the following files and subdirectories (note that **bold** font indicates folders):

<BaseName>.cfg	Configuration file that contains the DRE 4 settings.
<BaseName>.exe	DRE 4 executable.
Start.sh	Start script for DRE 4.
Stop.sh	Stop script for DRE 4.
Uninstall.sh	Executable to uninstall DRE 4 from your computer.
examplecfgs	Folder that contains example configuration files and templates.
DRE3COMPATIBILITY	Folder that contains example configuration files for making DRE 4 compatible with DRE 3.
LANGUAGES	Folder that contains an example configuration file for using multiple languages with DRE 4.
SECURITY	Folder that contains an example configuration file for using security with DRE 4.
langfiles	Folder in which the data and text files are stored that the DRE requires to run multiple languages and convert encodings.
templates	Folder in which templates are stored that are required to display results.

2. Configuring the DRE

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

Displaying help on configuration settings

For details on the settings that the individual configuration file sections can contain and on how you can configure them, please refer to the DRE online help.

To display the online help

1. Issue the following command from your web browser:

   http://<host>:<port>/action=Help

   <host>

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

   <port>

   Enter the port number that client machines use to communicate with the DRE (this is specified by the **Port** setting in the DRE configuration file's **[Server]** section).

2. Click on the **config help** link in the top right-hand corner to display the configuration parameter help (by default the action command help is displayed).

   Note: the configuration file sections that each configuration parameter can be used in are listed under **Allowed in Sections**.

Note:

You can also generate configuration help without starting the DRE. Issue the following command from the command line to generate html files in your installation directory:

<AXE_installation_directory_path>axe.exe -help.

Modifying configuration parameter values

Entering Boolean values

For parameters that require Boolean settings the following settings are interchangeable

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

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

Entering string values

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

For example:

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

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

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

For example:

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

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

For example:

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

Applying modifications to the DRE's operation

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

Configuration file sections

The DRE4 / AXE configuration file comprises a number of sections, which represent different areas that can be configured. You can configure each area by setting configuration parameters for it. All available configuration parameters are detailed in the DRE4 / AXE HTML help, which you can generate in your installation directory by issuing the following command from the command line:

<AXE_installation_directory_path>axe.exe -help.

Note that the configuration file sections that each configuration parameter can be used in are listed under **Allowed in Sections**.

The configuration file can contain the following sections:

[License]

[Service]

[Server]

[TermCache]

[IndexCache]

[QueryCache]

[SectionBreaking]

[ConceptSummary]

[Paths]

[FieldProcessing]

[Properties]

[Schedule]

[Databases]

[Templates]

[Logging]

[LanguageTypes]

[Security]

Note: which of the above listed sections you require depends on which operations you want your DRE to carry out.

[License] section

The [License] section contains licensing details, which you should not change.

For example:

```
[License]
Holder=My Company
Key=01234567890
Operations=803|87sdhsdf9n94nmsf7oasda987w4yriasunfaasd
```

[Service] section

The [Service] section contains settings that determine which machines are permitted to use and control the DRE service.

For example:

```
[Service]
ServicePort=40010
ServiceControlClients=127.0.0.1
ServiceStatusClients=127.0.0.1
```

[Server] section

The [Server] section contains general settings for indexing and querying.

For example:

```
QueryClients=*.*.*.*
IndexClients=*.*.*.*
AdminClients=*.*.*.*
QueryPort=9000
IndexPort=9001
Port=9002
QueryThreads=4
MaximumThreads=4
MaxInputString=1000
RoundDays=1
InclusiveDates=1
DateFormatCSVs=DD/MM/YYYY,YYYY/MM/DD,YYYY-MM-DD
Default_xoptions=+v3+nocache
IDXFieldPrefix=DOCUMENT
KillDuplicates=*/DREREFERENCE,*/url
CombineFieldsCSVs=*/DREREFERENCE
DocumentDelimiterCSVs=*/SPEECH,*/article,*/body,*/DOCUMENT
```

[TermCache] section

The [TermCache] section contains settings that determine how much memory the DRE uses to cache query terms.

For example:

```
[TermCache]
TermCacheMaxSize=10240
```

[IndexCache] section

The [IndexCache] section contains settings that determine how much memory the DRE uses to cache data for indexing.

For example:

```
[IndexCache]
IndexCacheMaxSize=10240
```

[QueryCache] section

The [QueryCache] section contains settings that determine how much memory the DRE uses to cache queries and their results, and how long they are stored in the cache.

For example:

```
[QueryCache]
Queries=200
ExpirySeconds=1000
```

[SectionBreaking] section

The [SectionBreaking] section contains settings that determine the size of the sections that documents are broken up into before they are indexed.

For example:

```
[SectionBreaking]
MinFieldLength=80
MaxSectionLength=2000
```

[ConceptSummary] section

The [ConceptSummary] section contains settings that determine the length of sentences that can be used in documents' concept summaries.

For example:

```
[ConceptSummary]
MinWordsPerSentence=10
MinWordsPerSentence=30
```

[Paths] section

The [Paths] section contains settings that allow you to split the database multiple partitions.

For example:

```
[PATHS]
DyntermPath=./dynterm
NodetablePath=./nodetable
RefIndexPath=./refindex
MainPath=./main
LogPath=./logs
NumericPath=./numeric
```

[FieldProcessing] section

The [FieldProcesssing] section lists the processes that you want to apply to fields, and contains a section for each of the processes, in which you define the process.

For example:

```
[FieldProcessing]
Number=4
0=IndexFields
1=IndexAndWeightHigher
2=SectionBreakFields

[IndexFields]
Property=Index
PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE
```

```
[IndexAndWeightHigher]
Property=IndexWeight
PropertyFieldCSVs=*/SUMMARIES

[SectionBreakFields]
Property=Section
PropertyFieldCSVs=*/DRESECTION
```

[Properties] section

The [Properties] section lists the properties that you have created within the sections of the individual processes that you want to be applied to fields, and contains a section for each of the properties, in which you set configuration parameters that are applied to associated fields.

For example:

```
[Properties]
0=Index
1=IndexWeight
2=Section
3=Date

[Index]
Index=TRUE

[IndexWeight]
Index=TRUE
Weight=2

[Section]
SectionBreakType=TRUE

[Date]
DateType=TRUE
```

[Schedule] section

The [Schedule] section contains settings that allow you to schedule when the DRE is compacted and when documents are expired from databases.

For example:

```
[Schedule]
Compact=true
Expire=true
CompactTime=00:00
CompactInterval=672
ExpireTime=00:00
ExpireInterval=24
```

[Databases] section

The [Databases] section lists the databases in which the DRE stores its data. You can set up a section for each of the databases, in which you specify settings that only apply to this database.

For example:

```
[Databases]
NumDBs=3
0=Archive
1=News
2=Misc

[Archive]
Expire=true
ExpireTime=00:00
ExpireInterval=604800
```

[Synonym] section

The [Synonym] section lists the settings that determine how the DRE handles synonym queries. A synonym query returns results which are conceptually similar to the query's terms and / or conceptually similar to the synonyms that are available for the query's terms.

For example:

```
[Synonym]
0=PC_Syn

[PC_Syn]
File=myfile.txt
MaxExpandLevel=1
ExpandLinks=true
```

Note: to be able to send synonym queries to the DRE, you need to set up a synonym file and add the Synonym action parameter to your query.

[Templates] section

The [Templates] section lists the templates that are required to output results in a manner that is compatible with non-ACI compatible applications. It contains a section for each of the listed templates, in which the templates' components are defined.

Note: you must include a [Template] section in the DRE's configuration file, even if you are only using ACI compatible applications.

For example:

```
[Templates]
0=results_template
1=content_template
DefaultResultsTemplate=results_template
DefaultContentTemplate=content_template

[results_template]
TemplateHeader=templates/resultsheader.txt
TemplateBody=templates/resultsbody.txt
TemplateFooter=templates/resultsfooter.txt
TemplateMimeType=text/plain

[content_template]
TemplateHeader=templates/contentheader.txt
TemplateBody=templates/contentbody.txt
TemplateFooter=templates/contentfooter.txt
```

[Logging] section

The [Logging] section lists the logging streams that you want to set up in order to create separate log files for different log message types (query, index and application). It contains a section for each of the listed logging streams, in which you can configure the settings that determine how each stream is logged.

For example:

```
[Logging]
0=INDEX_LOG_STREAM
1=QUERY_LOG_STREAM
2=APP_LOG_STREAM

[INDEX_LOG_STREAM]
LogFile=index.log
LogHistorySize=55
LogTime=true
LogEcho=true
MaxLogSizeKbs=1024
LogTypeCSVs=index
LogLevel=full
oldlogfileaction=datestamp

[QUERY_LOG_STREAM]
LogFile=query.log
LogHistorySize=50
LogTime=true
logecho=true
MaxLogSizeKbs=1024
LogTypeCSVs=query
LogLevel=full
oldlogfileaction=consecutive

[APP_LOG_STREAM]
LogFile=application.log
LogHistorySize=50
LogTime=true
LogEcho=true
MaxLogSizeKbs=1024
LogTypeCSVs=application
LogLevel=full
oldlogfileaction=datestamp
```

Note: all queries are truncated to 4000 characters in query logs.

[LanguageTypes] section

The [LanguagesTypes] section lists the language types that you want to use. It contains a section for each of the listed language types, in which you configure the settings that determine how each language type is handled.

For example:

```
[LanguageTypes]
DefaultLanguageType=english
LanguageDirectory=./langfiles
0=english

[english]
LanguageCode=1
Language=ENGLISH
Encoding=ascii
Propernames=1
Hyphenchars=-/
AugmentSeparators=
DiminishSeparators=
Stoplist=english.dat
Indexnumbers=1
Transliteration=FALSE
Stemming=TRUE
```

Note: use the [FieldProcessing] and [Properties] section to identify fields that determine the language of documents and the processes that should be applied to these fields or documents.

[Security] section

The [Security] section lists the security modules that you are using, and contains a section for each of the security modules, in which you can specify the settings that you want to apply to each module.

For example:

```
[Security]
SecurityInfoKeys=123,234,345,456
0=NT
1=Netware

[NT]
SecurityCode=1
Library=nt_security.dll
Type=AUTONOMY_SECURITY_V4_NT_MAPPED
ReferenceField=*/AUTONOMYMETADATA
Logging=1
DebugDecrypt=FALSE

[Netware]
SecurityCode=2
Library=netware_security.dll
Type=AUTONOMY_SECURITY_NETWARE_MAPPED
ReferenceField=*/AUTONOMYMETADATA
Logging=1
DebugDecrypt=TRUE
```

Note:

- use the [FieldProcessing] and [Properties] section to identify fields that determine the security type of documents and the processes that should be applied to these fields or documents.
- if you are running your DRE on a UNIX platform, you need to specify the LD_LIBRARY_PATH to ensure that the DRE can find the shared objects that it requires in order to implement security.

Configuring logging

You can set up logging streams, which allow you to create separate log files for different log message types (query, index and application).

Specify the name and location of the log files, the type of logging that should be performed (for example, full logging), if you want to display log messages on the console, the maximum size of log files and so on.

For example:

```
[Logging]
0=INDEX_LOG_STREAM
1=QUERY_LOG_STREAM
2=APP_LOG_STREAM

[INDEX_LOG_STREAM]
logfile=logs/index.log
loghistorysize=50
logtime=true
logecho=false
maxlogsizekbs=1024
logtypecsvs=index
loglevel=full

[QUERY_LOG_STREAM]
logfile=logs/query.log
loghistorysize=50
logtime=true
logecho=false
maxlogsizekbs=1024
logtypecsvs=query
loglevel=full

[APP_LOG_STREAM]
logfile=application.log
loghistorysize=50
logtime=true
logecho=false
maxlogsizekbs=1024
logtypecsvs=application
loglevel=full
```

Configuring security

You can apply specific security settings to documents that are indexed into the DRE by identifying fields in the documents that determine which security settings are appropriate to each of the documents (unless you want to specify the security property of a document every time you index a document by sending an additional parameter).

For details on the settings that the **[Security]** section can contain and on how you can configure them, please refer to the DRE 4 configuration help HTML file, which is located in your DRE 4 directory. To generate help, issue the following command on the command line:

```
<AXE_installation_directory_path>axe.exe -help
```

To set up automatic security application for documents:

1. In the DRE's configuration file's **[Security]** section:

   List the security types that you want to use, and specify the security keys that identify the DRE's security type.

   For example:

   ```
   [Security]
   SecurityInfoKeys=123,234,345,456
   0=NT
   1=Netware
   2=Notes
   3=Exchange
   ```

2. Define a section for each of the security types that you have defined (the section must have the same name as the security type), and specify appropriate settings for each security type in order to determine how the DRE handles this security type.

   For example:

   ```
   [NT]
   SecurityCode=1
   Library=nt_security.dll
   Type=AUTONOMY_SECURITY_V4_NT_MAPPED
   ReferenceField=*/AUTONOMYMETADATA

   [Netware]
   SecurityCode=2
   Library=netware_security.dll
   Type=AUTONOMY_SECURITY_NETWARE_MAPPED
   ReferenceField=*/AUTONOMYMETADATA
   ```

```
[Notes]
SecurityCode=3
Library=notes_security.dll
Type=AUTONOMY_SECURITY_V4_NOTES_MAPPED
ReferenceField=*/AUTONOMYMETADATA

[Exchange]
SecurityCode=4
Library=exchange_security.dll
Type=AUTONOMY_SECURITY_EXCHANGE_MAPPED
ReferenceField=*/AUTONOMYMETADATA
```

3. In the DRE's configuration file's **[FieldProcessing]** section:

   Set up processes that allow the DRE to recognize the security type of documents (unless you want to specify the security property of a document every time you index a document by sending an additional parameter). If you are using a version 4 security type (for example, AUTONOMY_SECURITY_V4_NOTES_MAPPED), you must include a process that defines how you want to handle metadata.

   For example:

   ```
   [FieldProcessing]
   Number=4
   0=DetectNT
   1=DetectNetware
   2=DetectNotes
   3=DetectExchange
   4=DefineMetaData
   ```

4. Create a section for each of the processes that you have listed, in which you create a property for the process (security properties always point to a defined security type). Identify the fields that you want to associate with the processes (when identifying the fields from which the DRE can read a document's language type you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to).

   You can use the **PropertyMatch** parameter to identify a specific value that fields must have in order to be processed.

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [DetectNT]
   Property=SetNTProperty
   PropertyFieldCSVs=*/DRESECURITYTYPE
   PropertyMatch=*nt
   ```

```
[DetectNetware]
Property=SetNetwareProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*netware

[DetectNotes]
Property=SetNotesProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*notes

[DetectExchange]
Property=SetExchangeProperty
PropertyFieldCSVs=*/DRESECURITYTYPE
PropertyMatch=*exchange

[DefineMetaData]
Property=HideMetaData
PropertyFieldCSVs=*/AUTONOMYMETADATA
```

5. List all the Properties that you have created in a **[Properties]** section.

   For example:

   ```
   [Properties]
   0=SetNTProperty
   1=SetNetwareProperty
   2=SetNotesProperty
   3=SetExchangeProperty
   4=HideMetaData
   ```

6. Create a section for each of the Properties and specify appropriate configuration settings for each. These configuration parameters define the processes that are applied to all the fields (or all documents that contain the fields) that you have previously associated with the processes.

   Note: if you are using a version 4 security type (for example, AUTONOMY_SECURITY_V4_NOTES_MAPPED), you must set **ACLType** to **true** in the section that sets up how the DRE handles metadata, in order to implement optimized security.

   ```
   [SetNTProperty]
   SecurityType=NT

   [SetNetwareProperty]
   SecurityType=Netware
   ```

```
[SetNotesProperty]
SecurityType=Notes
[SetExchangeProperty]
SecurityType=Exchange

[HideMetaData]
HiddenType=true
ACLType=true
```

Configuring languages

By default the DRE is set up to process English documents. You can however run the DRE in any of the languages listed in the **Running the DRE in multiple languages** chapter by modifying the DRE's configuration file.

For details on the settings that the **[LanguageTypes]** sections can contain and on how you can configure them, please refer to the DRE 4 configuration help HTML file, which is located in your DRE 4 directory. To generate help, issue the following command on the command line:

<AXE_installation_directory_path>axe.exe -help

To run the DRE in multiple languages:

1. In the DRE's configuration file's **[LanguageTypes]** section:

   Specify language types for each of the language and encoding combinations that you want to use.

   For example:

   ```
   [LanguageTypes]
   0=arabic
   1=arabicISO
   2=english
   3=french
   ```

2. Define a section for each of the language types that you have defined (the section must have the same name as the language type), and specify appropriate settings for each language type in order to determine how the DRE handles this language type.

   For example:

   ```
   [arabic]
   LanguageCode=1
   Language=ARABIC
   Encoding=ARABIC

   [arabicISO]
   LanguageCode=2
   Language=ARABIC
   Encoding=ARABIC_ISO

   [english]
   LanguageCode=3
   Language=ENGLISH
   Encoding=ASCII
   ```

```
[french]
LanguageCode=4
Language=FRENCH
Encoding=ASCII
```

3. In the DRE's configuration file's **[FieldProcessing]** section:

   Set up processes that allow the DRE to recognize the language type of documents (unless you want to specify the language property of a document every time you index a document by sending an additional parameter).

   For example:

   ```
   [FieldProcessing]
   Number=4
   0=DetectArabic
   1=DetectArabicISO
   2=DetectEnglish
   3=DetectFrench
   ```

4. Create a section for each of the processes that you have listed, in which you create a property for the process (language properties always point to a defined language type). Identify the fields that you want to associate with the processes (when identifying the fields from which the DRE can read a document's language type you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to).

   You can use the **PropertyMatch** parameter to identify a specific value that fields must have in order to be processed.

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [DetectArabic]
   Property=SetArabicProperty
   PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
   PropertyMatch=arabic

   [DetectArabicISO]
   Property=SetArabicISOProperty
   PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
   PropertyMatch=arabicISO,ISOarab*

   [DetectEnglish]
   Property=SetEnglishProperty
   PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
   PropertyMatch=*eng*,uk,*british
   ```

```
[DetectFrench]
Property=SetFrenchProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*fre*,fran*
```

Enabling synonym queries

You can configure your DRE to use a synonym file in order to enable synonym queries. A synonym query returns results which are conceptually similar to the query's terms and / or conceptually similar to the synonyms that are available for the query's terms.

To be able to send synonym queries to the DRE, you need to:

1. **Configure the DRE to use a synonym file**
2. **Set up a synonym file**

For details on the settings that the **[Synonym]** sections can contain and on how you can configure them, please refer to the DRE 4 configuration help HTML file, which is located in your DRE 4 directory. To generate help, issue the following command on the command line:

<AXE_installation_directory_path>axe.exe -help

To configure the DRE to use a synonym file:

1. In the DRE configuration file's **[FieldProcessing]** section:

   Set up a synonym process. The synonym process will allow the DRE to determine when it should apply synonym settings. For example:

   ```
   [FieldProcessing]
   Number=1
   0=SynonymMatch
   ```

2. Create a section for the synonym process that you have listed, in which you create a property for the process (synonym properties always point to a defined synonym job). Identify the fields that you want to associate with the process (when identifying the fields that the DRE uses for synonym matching you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to).

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [SynonymMatch]
   Property=ApplySynonymMatch
   PropertyFieldCSVs=*/DRETITLE,*/DRECONTENT
   ```

   In this example the DRE will only return documents for synonym queries, if their DRETITLE or DRECONTENT field values match the query.

3. List the property that you have created in a **[Properties]** section.

   For example:

   ```
   [Properties]
   0=ApplySynonymMatch
   ```

4. Create a section for the property in which you set the **SynonymType** parameter to the name of the synonym job that specifies which settings the DRE should apply to synonym queries.

   ```
   [ApplySynonymMatch]
   SynonymType=Synonym_job
   ```

5. In the DRE's configuration file's **[Synonym]** section:

   List the synonym job whose settings you want to apply when a synonym query is sent to the DRE. (You can set up multiple jobs, however, normally you only require one).

   For example:

   ```
   [Synonym]
   0=Synonym_job
   ```

6. Define a section for your synonym job (the section must have the same name as the synonym job) in which you specify the settings that you want to apply to synonym queries.

   For example:

   ```
   [Synonym_job]
   File=animals.txt
   MaxExpandLevel=2
   ExpandLinks=true
   ```

To set up a synonym file:

1. Create a text file and save it in the DRE's installation directory using the **File** name you have specified in the DRE configuration file's [<Synonym_type>] section.
2. Create sections for each language type that you have defined in the DRE's configuration file.

   For example:

   ```
   [English_ASCII]
   [German_UTF8]
   ```
3. In each section create a line for each word that you want to list synonyms for.

   For example:

   ```
   [English_ASCII]
   cat
   dog
   [German_UTF8]
   Katze
   Hund
   ```
4. List synonym strings next to each word and save the file. Note that you must separate the word and each string with commas and that there must be no space before or after a comma.

   For example:

   ```
   [English_ASCII]
   cat,feline,grimalkin,moggy,mouser,puss,pussy,tabby
   dog,bitch,canine,cur,hound,man's best
   friend,mongrel,mutt,pooch,pup,puppy
   [German_UTF8]
   Katze,Mietze,Mietzekatze,Mietzekater,Kater,Mulle,Kätzchen
   Hund,Wau Wau,Hündin,Töle,Kläffer,Hündchen,Welpe
   ```

Storing the DRE's data files on multiple disks

If your DRE becomes too big to be stored on one disk (as the terms, references, content and so on that it stores increase in size), you can store its data files across multiple partitions within your PC in order to gain space.

For example:

```
[PATHS]
DyntermPath=C:\autonomy\axe\dynterm
NodetablePath=D:\autonomy\axe\nodetable
RefIndexPath=E:\autonomy\axe\refindex
MainPath=F:\autonomy\axe\main
TagPath=.G:\autonomy\axe\tagindex
```

Configuring databases

You can specify the fields from which you want the DRE to read into which database it should index a file.

For example:

```
[DatabaseFields]
Property=Database
PropertyFieldCSVs=*/DREDBNAME,*/DB,*/Database

[Properties]
0=Database

[Database]
DatabaseType=TRUE
```

Fields

Storing and indexing fields

When documents are indexed into DRE 4, the DRE by default automatically stores all the fields that the documents contain.

To speed up the indexing process and optimize memory usage, you can prevent the DRE from storing fields that you do not want to use by setting up the **CantHaveCSVs** parameter in your DRE configuration file, or by adding the **CantHaveFields** parameter to your DREADD or DREADDDATA indexing command.

You can also explicitly index fields into the DRE (using the DRE's configuration file) in order to optimize the query process when you restrict queries using these fields. Index fields should hold data that is particularly significant to you (for example the title of the document), and that you are likely to use frequently in order to restrict queries.

Note: you should not index fields that you are unlikely to use or whose content is irrelevant to you (for example, NumPages or DRESection). Indexing all fields in documents could potentially slow down the indexing process, increase memory usage and disk requirements.

To explicitly index fields into the DRE:

1. List an indexing process in the **[FieldProcessing]** section.

   For example:

   ```
   [FieldProcessing]
   Number=3
   0=MyFirstProcess
   1=MySecondProcess
   2=IndexingFields
   ```

2. Create a section for the indexing process, in which you create a property for the process (a property is later defined by one or more applicable configuration parameters). Identify the fields that you want to associate with the processes.

   You can use the **PropertyMatch** parameter to identify a specific value that fields must have in order to be processed.

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [MyFirstProcess]
   Property=MyFirstProperty
   PropertyFieldCSVs=*/MyField,*/MySecondField
   PropertyMatch=*myString*
   ```

```
[MySecondProcess]
Property=MySecondProperty
PropertyFieldCSVs=*/MyOtherField,*/MyOtherSecondField

[IndexingFields]
Property=IndexFields
PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE
```

3. List the properties that you have created in a **[Properties]** section.

   For example:

```
[Properties]
0=MyFirstProperty
1=MySecondProperty
2=IndexFields
```

4. Create a section for your indexing Property in which you set the Index parameter to true.

   For example:

```
[MyFirstProperty]
HiddenType=true

[MySecondProperty]
Index=true

[IndexFields]
Index=true
```

Processing fields and documents that contain specific fields

The **[FieldProcessing]** section in the DRE's configuration file allows you to identify particular fields in documents and, depending on their value, apply any type of processing to them or the document that contains them during the indexing process.

This means that you can apply multiple processes to documents without having to set up a configuration section for each process combination.

Note: when identifying fields you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to. If you just specify the **FieldName**, the DRE automatically adds a ***/** to it

To apply processes to specific fields or documents that contain specific fields:

1. List the processes that you want to apply to fields in the **[FieldProcessing]** section.

   For example:

   ```
   [FieldProcessing]
   Number=4
   0=MyFirstProcess
   1=IndexFields
   2=MyCombinedProcess
   3=IndexAndWeightHigher
   ```

2. Create a section for each of the processes that you have listed, in which you create a property for the process (a property is later defined by one or more applicable configuration parameters). Identify the fields that you want to associate with the processes.

   You can use the **PropertyMatch** parameter to identify a specific value that fields must have in order to be processed.

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [MyFirstProcess]
   Property=MyFirstProperty
   PropertyFieldCSVs=*/MyField,*/MySecondField
   PropertyMatch=*myString*

   [IndexFields]
   Property=MySecondProperty
   PropertyFieldCSVs=*/DRECONTENT,*/DRETITLE
   ```

```
[MyCombinedProcess]
Property=MyCombinedProperty
PropertyFieldCSVs=*/MyField,*/MySecondField

[IndexAndWeightHigher]
Property=IndexHigherWeight
PropertyFieldCSVs=*/SUMMARIES
```

3. List all the properties that you have created in a **[Properties]** section.

   For example:

   ```
   [Properties]
   0=MyFirstProperty
   1=MySecondProperty
   2=MyCombinedProperty
   3=IndexHigherWeight
   ```

4. Create a section for each of the properties and specify appropriate configuration settings for each. These configuration parameters define the processes that are applied to all the fields (or all documents that contain the fields) that you have previously associated with the processes.

   For example:

   ```
   [MyFirstProperty]
   HiddenType=true

   [MySecondProperty]
   Index=true

   [MyCombinedProperty]
   DateType=true
   Index=true

   [IndexHigherWeight]
   Index=true
   Weight=2
   ```

Note: for details on available configuration settings please refer to the DRE 4 configuration help HTML file, which is located in your DRE 4 directory. To generate help html files in your installation directory, issue the following command on the command line:

<AXE_installation_directory_path>axe.exe -help

Example:

```
[FieldProcessing]
Number=6
0=IndexFields
1=IndexAndWeightHigher
2=SectionBreakFields
3=DateFields
4=DatabaseFields
5=SetReferenceFields

[IndexFields]
// Controls which fields are indexed
Property=Index
PropertyFieldCSVs=*/*

[IndexAndWeightHigher]
// Fields which are indexed with a weight
Property=IndexWeight
PropertyFieldCSVs=*/SUMMARIES

[SectionBreakFields]
// Field containing document section number
Property=Section
PropertyFieldCSVs=*/DRESECTION

[DateFields]
// Fields containing the document date
Property=Date
PropertyFieldCSVs=*/DREDATE,*/harvest_time

[DatabaseFields]
// CSV of field names that define the document's database
Property=Database
PropertyFieldCSVs=*/DREDBNAME

[SetReferenceFields]
//CSV of fields that define the document's URL
Property=Reference
PropertyFieldCSVs=*/DREREFERENCE,*/DRETITLE
```

```
//---------------------------Properties-------------------------
------//
[Properties]
0=Index
1=IndexWeight
2=Section
3=Date
4=Database
5=Reference

[Index]
Index=TRUE

[IndexWeight]
Index=TRUE
Weight=2

[Section]
SectionBreakType=TRUE

[Date]
DateType=TRUE

[Database]
DatabaseType=TRUE

[Reference]
ReferenceType=TRUE
TrimSpaces=TRUE
```

Using fields to restrict queries: Field specifiers

You can restrict the results that a **Query**, **Suggest** or **SuggestOnText** action produces by specifying fields and field values that documents must contain in order to be returned as results. Field specifiers determine how documents match fields and field values in order to be returned as results.

As part of a query, field specifiers take the following format:

http://<IPaddress>:<port>/action=<ActionName>&FieldText=<Boolean_expression of (term:field) AND(specifier{arguments}:field)>

Please refer to the online help for a list of available field specifiers and details of how to use them.

You can display the online help by issuing the following command from your web browser:

http://<host>:<port>/action=Help

<host>

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

<port>

Enter the port number that client machines use to communicate with the DRE (this is specified by the **Port** setting in the DRE configuration file's **[Server]** section).

Note:

- When identifying fields you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to. If you just specify the **FieldName**, the DRE automatically adds a ***/** to it.
- All string matching is case insensitive.
- Strings can contain punctuation (except curly brackets), which means that if you want to match a string that contains html with DRE content, you may need to escape the html to avoid confusion with "&" and so on.

  If you want to match a string that contains a comma, you need to escape the comma with a backslash, otherwise the DRE reads it as a separator.

Returning specific fields for query results

When you send a **Query**, **Suggest** or **SuggestOnText** ACI action command to the DRE it returns by default all fields that you have set up as print fields (by setting the fields' **PrintType** property to **true** in the DRE configuration file).

Alternatively, you can add the **Print** parameter to your action command, which allows you to return all XML fields, all reference fields, all date fields, all explicitly indexed fields, the content of the index fields as a single text field,.

Note: for details on available actions and action parameters, please refer to the help on DRE action commands, which you can display by sending the following command from your web browser:

http://<host>:<port>/action=Help

<host>

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

<port>

Enter the port number that client machines use to communicate with the DRE (this is specified by the **Port** setting in the DRE configuration file's **[Server]** section).

Setting up Highlight fields

When you execute a Query, Suggest or SuggestOnText action command, you can highlight sentences or words in the results that are related to the terms in the query (or the terms in the text or document that you are suggesting on).

The DRE checks which fields highlighting applies to and then highlights all sentences or words that are based on the terms in the results that it returns.

To apply highlighting to fields:

1. List a highlighting process in the **[FieldProcessing]** section.

   For example:

   ```
   [FieldProcessing]
   Number=2
   0=MyFirstProcess
   1=HighlightFields
   ```

2. Create a section for the highlighting process that you have listed, in which you create a property for the process (a property is later defined by one or more applicable configuration parameters). Identify the fields that you want to associate with the processes.

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [HighlightFields]
   Property=Highlight
   PropertyFieldCSVs=*/DRETITLE,*/DRECONTENT
   ```

3. List the property that you have created in the **[Properties]** section.

   For example:

   ```
   [Properties]
   0=MyFirstProperty
   1=Highlight
   ```

5. Create a section for the property in which you set the **HighlightingType** parameter to **true**. This enables the highlighting of all matched terms that are contained in the associated **PropertyFieldCSVs** fields.

   For example:

   ```
   [Highlight]
   HighlightType=true
   ```

Setting up memory mapping to speed up numerical queries

You can configure the DRE to identify fields that contain numerical values. When these fields are queried, the DRE stores them in memory so it can quickly return the field.

The DRE stores fields in memory in an intelligent way, determining by how often fields are queried which value ranges are most likely to be queried. This means that numerical values that are not queried are stored on disk and don't take up memory. If the query behavior changes the DRE automatically adjusts which value ranges it stores in memory.

To set up memory mapping:

1. List a process that identifies numerical fields in the **[FieldProcessing]** section.

   For example:

   ```
   [FieldProcessing]
   Number=2
   0=MyFirstProcess
   1=PriceFields
   ```

2. Create a section for the process that you have listed, in which you create a property for it (a property is later defined by one or more applicable configuration parameters). Identify the fields that you want to associate with the process.

   Note: the properties that you create must not have the same name as processes.

   For example:

   ```
   [PriceFields]
   Property=Price
   PropertyFieldCSVs=*/PRICE
   ```

3. List the property that you have created in the **[Properties]** section.

   For example:

   ```
   [Properties]
   0=MyFirstProperty
   1=Price
   ```

6. Create a section for the property in which you set the **NumericType** parameter to **true**. This enables the DRE to memory map the associated **PropertyFieldCSVs** fields.

   For example:

   ```
   [Price]
   NumericType=true
   ```

7. Restart the DRE, so your new settings can take effect.

If you now send query for a specific value that is stored in the */PRICE field, the DRE will memory map the range that this value is in, so it can return results more quickly next time a value that lies in this range is queried.

Examples:

http://12.3.4.56:4000/action=Query&FieldText=NRANGE{50,100}:*/PRICE

A document's **PRICE** field must contain a number between **50** and **100** (including decimal numbers) for this document to be returned.

http://12.3.4.56:4000/action=Query&Text=computer&Sort=*/PRICE:numincreasing

The results that the DRE returns for the query are sorted according to the values they their **PRICE** fields contain. The results whose **PRICE** field contains the smallest value is listed first, followed by results with increasing values in the **PRICE** field.

3. Importing and indexing data into the DRE

Before you can index files into a DRE they must be imported into XML or IDX file format. You can import files into XML or IDX format:

using a Connector

The Autonomy Connectors (for example, AutoIndexer, HTTPFetch, Oracle Fetch and so on) allow you to retrieve documents from different repositories and import them into IDX file format only. Please refer to the appropriate Connector manual for further information on how to import documents.

manually

You can create a text file in XML or IDX format, which contains the information that you want to index into your DRE in specific DRE fields.

using the DRE Administration dialog

If you are running on a Windows NT or 2000 platform, you can use the DRE Administration dialog to import documents into XML or IDX file format. Please refer to **Appendix C** for further details on the DRE Administration dialog.

Once documents have been imported into XML or IDX file format, you can index them into a DRE:

using a Connector

The Autonomy Connectors allow you to index the IDX files that they have created into the DRE that they connect to. Please refer to the appropriate Connector manual for further information on how to index documents.

directly

You can index XML and IDX files into a DRE using an HTTP request that you can issue from your web browser.

using the DRE Administration dialog

If you are running on a Windows NT or 2000 platform, you can use the DRE Administration dialog to index XML and IDX documents into a DRE. Please refer to **Appendix C** for further details on the DRE Administration dialog.

Note: depending on where the data that the DRE indexes is located, the indexing process takes place in the following order:

DRE indexes a locally accessible file:

1. The DRE receives a filename.
2. The DRE opens the file and reads the data.
3. The indexing process takes place.

DRE receives data over the indexing port:

1. The DRE receives a stream of data over the port.
2. The DRE saves the data locally.
3. The DRE opens the file and reads the data.
4. The indexing process takes place

DREADD: directly indexing IDX and XML files

http://<host>:<port>/DREADD?<mandatory_parameter>&<optional_parameters>

The **DREADD** command allows you to index IDX or XML files that are located on the same machine as the DRE directly into a DRE. Note that parameters that you use with **DREADD** override any equivalent settings that you may have specified in the DRE's configuration file.

Command parameters:

Mandatory:	**<file_name>**
	or
	<path>
Optional:	**ACLFields**=<ACL_fields>
	CantHaveFields=<forbidden_fields>
	DatabaseFields=<database_fields>
	DateFields=<date_fields>
	Delete
	DocumentDelimiters=<doc_delimiters>
	DocumentFormat=<doc_format>
	DREDbName=<database_name>
	ExpiryDateFields=<expiry_date_fields>
	FlattenIndexFields=<fields>
	IDXFieldPrefix=<prefix>
	IndexFields=<index_fields>
	KillDuplicates=<kill_duplicates_option>
	LanguageFields=<language_fields>
	LanguageType=<language_type>
	MustHaveFields=<required_fields>
	SectionFields=<section_fields>

SecurityFields=<security_fields>

SecurityType=<security_type>

TitleFields=<title_fields>

Note:

When identifying fields you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to. If you just specify the **FieldName**, the DRE automatically adds a ***/** to it.

<host>

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

<port>

Enter the number of the index port by which files are indexed into the DRE.

<file_name>

The IDX or XML file that you want to index.

<path>

The full path to the IDX or XML file that you want to index.

<optional_parameters>

You can enter one or more of the following parameters (note that you must separate individual parameters with an ampersand):

ACLFields=<ACL_fields>

Allows you to specify the fields in the document from which you want the DRE to read ACLs (Access Control Lists).

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&ACLFields=*/AUTONOMYMETADATA

In this example, the DRE reads ACLs from any fields that are called **AUTONOMYMETADATA**.

CantHaveFields=<forbidden_fields>

Allows you to specify the fields in XML documents that are discarded before the documents is indexed. By default all fields are stored in the DRE.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&CantHaveFields=*/StandardHeader

In this example, any **StandardHeader** fields that a document contains are discarded before the document is indexed.

DatabaseFields=<database_fields>

Allows you to specify the fields in the document that contain the name of the database in which you want the document to be stored.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&DatabaseFields=Document/DREDBName,*/myDB

In this example, the DRE indexes the document into the database with the name that is contained in any **DREDBName** field below the **Document** level and with the name that is contained in any fields called **myDB**.

DateFields=<date_fields>

Allows you to specify the fields in the document from which you want the DRE to read the document's date.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&DateFields=Document/DREDate,*/myDocDate

In this example, the DRE extracts dates from any fields that are called **DREDate** that are contained below the **Document** level and from any fields that are called **myDocdate**.

Delete

The DRE deletes the file that you are indexing after it has been indexed.

DocumentDelimiters=<doc_delimiters>

Allows you to specify the fields in a file that indicate the beginning and end of a document, so the documents are indexed individually. You should only have one document level per XML schema.

For example:

&DocumentDelimiters=*/DOCUMENT,*/SPEECH

In this example, the beginning and end of individual documents in a file is marked by opening and closing **DOCUMENT** and **SPEECH** tags.

DocumentFormat=<doc_format>

If a document that you are indexing has an ambiguous format that the DRE cannot easily identify as XML or IDX, **DocumentFormat** allows you to specify the format of the file. Enter **XML** or **IDX**.

DREDbName=<database_name>

Allows you to specify the DRE database into which you want the document to be indexed.

ExpiryDateFields=<expiry_date_fields>

Allows you to specify the fields in the document that contain the expiry date of the document (that is the date when the document is deleted, unless you have set **ExpireIntoDatabase** in the DRE's configuration file to move the document into another database).

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&ExpiryDateFields=Document/DREExpiryDate,*/myExpiryDate

In this example, the DRE reads the expiry date from any **DREExpiryDate** field below the **Document** level and from any fields called **myExpiryDate**.

FlattenIndexFields=<fields>

Allows you to specify the fields in a hierarchically structured document whose content you want to index as one level.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
    <documents>
        <article id="_21498602">
        <url>http://example.com/21490.html</url>
        <hltext_display>The history of pharmacogenetics </hltext_display>
        <source>Science Online</source>
        <media_type>text</media_type>
        <subject>
            <text>The prologue to pharmacogenetics began to play out around 1850 and
            spanned some 60 years into the 1900s.</text>
            <text>In 1953, the molecular basis of heredity, the double helix of DNA, was
            described.</text>
        </subject>
        <valid_time>Jul 13 2001  5:00AM</valid_time>
        </article>
    </documents>
```

If you specify **FlattenIndexFields=*/subject**, and index the above, any content that a **subject** field or a field within a **subject** field comprises is indexed as this **subject** field's content.

If you now query for a particular term in the **subject** field that is actually contained in a level below the **subject** field, for example the term "pharmacogenetics", the above text is returned. If you had not flattened the **subject** field the query would fail, as the subject field itself does not contain this term.

IDXFieldPrefix=<prefix>

When you index an IDX file it is transformed into XML by placing it under the **Document** subtree (each of the IDX file's fields is prefixed with **Document**, so that a simple XML hierarchy is constructed). If you don't want this subtree to be called **Document**, **IDXFieldPrefix** allows you to specify an alternative name.

IndexFields=<index_fields>

Allows you to specify the fields in the document that you want to index explicitly into the DRE. Indexing fields explicitly optimizes the query process when you restrict queries using these fields. Index fields should hold data that is particularly significant to you (for example the title of the document), and that you are likely to use frequently in order to restrict queries.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&IndexFields=*/DRECONTENT,*/DRETITLE

In this example, the **DRECONTENT** and **DRETITLE** field in documents are explicitly indexed into the DRE.

KillDuplicates=<kill_duplicates_option>

You can enter the following option to determine how the DRE handles duplicate text. Note that if you postfix any of these options with **2**, the **KillDuplicates** process is applied to all DRE databases (rather than just the database into which the IDX or XML file is indexed) :

REFERENCE

If a document is indexed that has the same **DREREFERENCE** as a document that the DRE already contains, the DRE deletes the document that it already contains and replaces it with the new one.

REFERENCEMATCHNN

If a document is indexed whose content is more than **NN** percent similar to the content of a document that the DRE already contains, the DRE deletes the document that it already contains and replaces it with the new one.

<FieldName>

If a document is indexed that contains a _**<FieldName>**_ field, which has the same content as the _**<FieldName>**_ field in a document that the DRE already contains, the DRE deletes the document that it already contains and replaces it with the new one.

LanguageFields=<language_fields>

Allows you to specify the fields in the document that contain the language type of the document.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&LanguageFields=Document/DRELanguageType,*/myLanguageType

In this example, the DRE reads the language type of documents from any **DRESection** field below the **Document** level and any **mySection** fields.

LanguageType=<language_type>

Allows you to specify the language type of documents (for example, if the document does not contain fields from which the DRE can read the language type of the document).

For example:

&LanguageType=myEnglish

In this example, the file is indexed with the language type **myEnglish**. The way the DRE handles this language type is determined by the way it has been defined in the DRE's configuration file (that is by the settings that you have associated with this language type in the configuration file).

MustHaveFields=<required_fields>

Allows you to specify the fields in a document (IDX only) that are stored in the DRE. By default all fields are stored in the DRE. Document fields that are not listed are discarded which means that they cannot be queried or printed.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&MustHaveFields=*/DRECONTENT,*/DRETITLE

In this example, the DRE only stores a document's **DRECONTENT** and **DRETITLE** fields.

SectionFields=<section_fields>

Allows you to specify the fields in the document that indicate the start of a new section in the document.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&SectionFields=Document/DRESection,*/mySection

In this example, any **DRESection** field below the **Document** level and any **mySection** fields indicate the start of a new section.

SecurityFields=<security_fields>

Allows you to specify the fields in the document that contain the security type of the document.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&SecurityFields=Document/DRESecurity,*/mySecurity

In this example, the DRE reads the security type of documents from any **DRESecurity** field below the **Document** level and any **mySecurity** fields.

SecurityType=<security_type>

Allows you to specify the security type of documents (for example, if the document does not contain fields from which the DRE can read the security type of the document).

For example:

&SecurityType=mySecurity

In this example, the file is indexed with the security type **mySecurity**. The way the DRE handles this security type is determined by the way it has been defined in the DRE's configuration file (that is by the settings that you have associated with this security type in the configuration file).

TitleFields=<title_fields>

Allows you to specify the field in the document from which you want the DRE to read the document's title.

For example:

&TitleFields=*/DRETITLE

In this example, the DRE reads a document's title from its **DRETITLE** field.

DREADDDATA: indexing data over a socket

DREADDDATA?<data>#DREENDDATA&<optional_parameters>

Note: This command requires a POST request method

The **DREADDDATA?…#DREENDDATA** command allows you to index data over a socket into a DRE. Note that parameters that you use with **DREADDDATA?…#DREENDDATA** override any equivalent settings that you may have specified in the DRE's configuration file.

Command parameters:

Mandatory:	**<data>**
Optional:	**KillDuplicates**=<kill_duplicates_option>
	Delete
	DocumentFormat=<doc_format>
	DREDbName=<database_name>
	IDXFieldPrefix=<prefix>
	DateFields=<date_fields>
	DatabaseFields=<database_fields>
	ExpiryDateFields=<expiry_date_fields>
	SectionFields=<section_fields>
	CantHaveFields=<forbidden_fields>
	FlattenIndexFields=<fields>
	SecurityFields=<security_fields>
	SecurityType=<security_type>
	LanguageFields=<language_fields>
	LanguageType=<language_type>
	DocumentDelimiters=<doc_delimiters>

<data>

The data that you want to index.

<optional_parameters>

You can enter one or more of the following parameters (note that you must separate individual parameters with an ampersand):

KillDuplicates=<kill_duplicates_option>

You can enter the following kill duplicates option to determine how the DRE handles duplicate text:

REFERENCE

If data is indexed that has the same **DREREFERENCE** as a document that the DRE already contains, the DRE deletes the document that it already contains and replaces it with the new one.

REFERENCEMATCHNN

If data is indexed that is more than **NN** percent similar to the content of a document that the DRE already contains, the DRE deletes the document that it already contains and replaces it with the new one.

<*/FieldName>

If data is indexed that contains a **<*/FieldName>** field, which has the same content as the **<*/FieldName>** field in a document that the DRE already contains, the DRE deletes the document that it already contains and replaces it with the new one.

Note: when identifying fields you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to. If you just specify the **FieldName**, the DRE automatically adds a ***/** to it.

Delete

The DRE deletes the data after it has been indexed.

DocumentFormat=<doc_format>

If data that you are indexing has an ambiguous format that the DRE cannot easily identify as XML or IDX, **DocumentFormat** allows you to specify the format of the data. Enter **XML** or **IDX**.

DREDbName=<database_name>

Allows you to specify the DRE database into which you want the data to be indexed.

IDXFieldPrefix=<prefix>

When you index IDX data it is transformed into XML by placing it under the **Document** subtree (each of the IDX file's fields is prefixed with **Document**, so that a simple XML hierarchy is constructed). If you don't want this subtree to be called **Document**, **IDXFieldPrefix** allows you to specify an alternative name.

DateFields=<date_fields>

Allows you to specify the fields in the data from which you the DRE to extract the date.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&DateFields=Document/DREDate,*/myDocDate

In this example, the DRE extracts dates from any fields that are called **DREDate** that are contained below the **Document** level and from any fields that are called **myDocdate**.

DatabaseFields=<database_fields>

Allows you to specify the fields in the data that contain the name of the database in which you want the data to be stored.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&DatabaseFields=Document/DREDBName,*/myDB

In this example, the DRE indexes the data into the database with the name that is contained in any **DREDBName** field below the **Document** level and with the name that is contained in any fields called **myDB**.

ExpiryDateFields=<expiry_date_fields>

Allows you to specify the fields in the data that contain the expiry date of the data (that is the date when the data is deleted, unless you have set **ExpireIntoDatabase** in the DRE's configuration file to move the data to another database).

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&ExpiryDateFields=Document/DREExpiryDate,*/myExpiryDate

In this example, the DRE reads the expiry date from any **DREExpiryDate** field below the **Document** level and from any fields called **myExpiryDate**.

SectionFields=<section_fields>

Allows you to specify the fields in the data that indicate the start of a new section in the data.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&SectionFields=Document/DRESection,*/mySection

In this example, any **DRESection** field below the **Document** level and any **mySection** fields indicate the start of a new section.

CantHaveFields=<forbidden_fields>

Allows you to specify the fields in XML data that are discarded before the data is indexed.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&CantHaveFields=*/StandardHeader

In this example, any **StandardHeader** field that a document contains is discarded before the data is indexed.

FlattenIndexFields=<fields>

Allows you to specify the fields in a hierarchically structured data whose content you want to index as one level.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <documents>
    <article id="_21498602">
    <url>http://example.com/21490.html</url>
    <hltext_display>The history of pharmacogenetics </hltext_display>
    <source>Science Online</source>
    <media_type>text</media_type>
    <subject>
      <text>The prologue to pharmacogenetics began to play out around 1850 and
      spanned some 60 years into the 1900s.</text>
      <text>In 1953, the molecular basis of heredity, the double helix of DNA, was
      described.</text>
    </subject>
    <valid_time>Jul 13 2001  5:00AM</valid_time>
```

```
        </article>
    </documents>
```

If you specify **FlattenIndexFields=*/subject**, and index the above, any content that a **subject** field or a field within a **subject** field comprises is indexed as this **subject** field's content.

If you now query for a particular term in the **subject** field that is actually contained in a level below the **subject** field, for example the term "pharmacogenetics", the above text is returned. If you had not flattened the **subject** field the query would fail, as the subject field itself does not contain this term.

SecurityFields=<security_fields>

Allows you to specify the fields in the data that contain the security type of the data.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&SecurityFields=Document/DRESecurity,*/mySecurity

In this example, the DRE reads the security type of data from any **DRESecurity** field below the **Document** level and any **mySecurity** fields.

SecurityType=<security_type>

Allows you to specify the security type of documents (for example, if the document does not contain fields from which the DRE can read the security type of the document).

For example:

&SecurityType=mySecurity

In this example, the file is indexed with the security type **mySecurity**. The way the DRE handles this security type is determined by the way it has been defined in the DRE's configuration file (that is by the settings that you have associated with this security type in the configuration file).

LanguageFields=<language_fields>

Allows you to specify the fields in the data that contain the language type of the document.

Note that if you want to specify multiple fields you must separate them with commas (there must be no space before or after a comma). You can use wildcards.

For example:

&LanguageFields=Document/DRELanguageType,*/myLanguageType

In this example, the DRE reads the language type of data from any **DRESection** field below the **Document** level and any **mySection** fields.

LanguageType=<language_type>

Allows you to specify the language type of data (for example, if the data does not contain fields from which the DRE can read the language type of the data).

For example:

&LanguageType=myEnglish

In this example, the data is indexed with the language type **myEnglish**. The way the DRE handles this language type is determined by the way it has been defined in the DRE's configuration file (that is by the settings that you have associated with this language type in the configuration file).

DocumentDelimiters=<doc_delimiters>

Allows you to specify the tags in data that indicate the beginning and end of a document, so the documents are indexed individually. You should only have one document level per XML schema.

For example:

&DocumentDelimiters=*/DOCUMENT,*/SPEECH

In this example, the beginning and end of individual documents in the data is marked by opening and closing **DOCUMENT** and **SPEECH** tags.

Checking if the indexing process was successful

You can check if the indexing of data into the DRE has been successful by running your web browser and entering the following:

http://<IPAddress>:<Port>/action=IndexerGetStatus

<IPAddress>	Enter your computer's IP address.
<Port>	Enter the number of the port that is used to send commands to the ACI Server.

The **IndexerGetStatus** command displays the status of the DRE's index queue:

-1	**Finished**	The indexing process is finished.
-2	**Out of disk space**	The DRE ran out of disk space before the indexing process could be completed.
-3	**File not found**	The index file could not be found.
-4	**Database not found**	The database into which you are trying to index could not be found.
-5	**Bad parameter**	The indexing command syntax is incorrect.
-6	**Database exists**	The database that you are trying to create already exists.
-7	**Queued**	The indexing command is queued and will be executed when all preceding indexing commands are finished.
-8	**Unavailable**	The DRE is about to shut down or indexing is paused.
-9	**Out of Memory**	The DRE ran out of memory before the indexing process could be completed.
-10	**Interrupted**	The indexing command was interrupted.
-11	**XML is not well formed**	Indexing failed because your XML is not well formed.

-12	**Retrying interrupted command**	The DRE is executing an indexing command that has previously been interrupted.
-13	**Backup in progress**	The DRE is performing a backup.
-14	**Max index size reached**	You have reached the maximum indexing size (the maximum indexing size depends on your license).
-15	**Max number of documents reached**	You have reached the maximum number of documents you can index (number of documents you can index depends on your license).
-16	**Index paused**	The indexing process has been paused.
-17	**Index restarted**	The indexing process has been restarted.
-18	**Index cancelled**	The indexing process has been cancelled.
-19	**Index out of file descriptors**	The DRE has run out of file descriptors.
-20	**Index languagetype not found**	The language type of the index data could not be found.
-21	**Index securitytype not found**	The security type of the index data could not be found.

Note: if the **IndexerGetStatus** command returns a positive number, this number indicates the percentage of the indexing queue that has been completed.

4. Querying the DRE

ACI query action command syntax

You can send query commands to the DRE from your web browser. The general syntax of these commands is as follows:

http://<host>:<port>/**action=<action>**&**<mandatory_parameters>**&**<optional_parameters>**

<host>

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

<port>

Enter the ACI port by which commands are sent to the DRE.

<action>

Enter the name of the action that you want the DRE to execute (for example, **Query**).

<mandatory_parameters>

Enter the parameters that the action that you have specified requires (not all actions require parameters).

<optional_parameters>

You can enter optional parameters for the action that you have specified (optional parameters are not available for all actions).

Note: you must separate individual parameters with an ampersand.

Displaying DRE action command help text

Enter the following command to display help on DRE action commands:

http://<host>:<port>/action=Help

<host>

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

<port>

Enter the ACI port by which commands are sent to the DRE.

Example:

http://12.3.4.56:4000/action=Help

This command uses port **4000** to request Help on action commands from the DRE, which is located on a machine with the IP address **12.3.4.56**.

5. Running the DRE in multiple languages

DRE 4 / AXE allows you to combine multiple languages in one DRE. To do this you need to:

- **Configure a multi-lingual DRE**

  Specify settings in the DRE's configuration file that allow it to recognize the language and encoding of documents (unless you want to define the language of a document every time that you are indexing a document) and that determine how it deals with the individual languages.

  Note: the internal DRE storage encoding is UTF8.

- **Index multi-lingual documents**

  If your DRE license enables Automatic Language Detection and you have set **AutoDetectLanguagesAtIndex** to **true** in the DRE's configuration file, the DRE automatically identifies the language and encoding of a document when it is indexed.

  If your license does not include this functionality, you have to specify the language and encoding of documents that you are indexing into the DRE or specify a field process in the DRE configuration file's [FieldProcessing] section, which allow the DRE to read the language of a document from one of its fields.

  Note:

  If you have Automatic Language Detection enabled and the DRE detects a document whose language type that has not been defined in its configuration file, the **DiscardUnconfiguredLanguagesAtIndex** configuration parameter determines how the DRE handles this document. If **DiscardUnconfiguredLanguagesAtIndex** is set to true, it discards the document. By default the DRE indexes the document using the default language type. It also logs a warning message in the index log, so that you can add an appropriate language type.

  If you have Automatic Language Detection enabled and a field process set up that reads a document's language from one of its fields, the DRE uses the field process rather than auto-detection to determine the document's language and encoding.

- **Query in multiple languages**

  Specify the language that your query is in, in order to return results in the same language or allow the DRE to return results in multiple languages.

Configuring a multi-lingual DRE

In order to run the DRE in multiple languages, you need to:

1. specify settings in the **[FieldProcessing]** section that allow the DRE to recognize which language type documents are written in (unless you want to specify the language property of a document every time you index a document by sending an additional parameter). Language properties always point to a defined language type.
2. define a **[LanguageTypes]** section in order to determine how the DRE handles each language type. For each language type you must specify **LanguageCode**, **Language** and **Encoding** settings.

Note: the internal DRE storage encoding is UTF8.

Setting up field processing in the DRE configuration file

Unless you want to specify the language property or type of a document every time you index a document, you need to define the fields in documents from which the DRE can read which language the documents are in.

Use the **[FieldProcessing]** section of the DRE's configuration file to define each language property that you want the DRE to be able to detect.

For example:

```
[FieldProcessing]
Number=6
0=DetectArabic
1=DetectArabicISO
2=DetectEnglish
3=DetectChSimplified
4=DetectChTraditional
5=DetectFrench
```

For each of the languages that you have defined in the **[FieldProcessing]** section, you need to define a section with the name of the respective language type. In this section you can then specify the fields that the DRE should look for and the values that those fields must have in order for the document to be recognized as a particular language type.

For example:

```
[DetectArabic]
Property=SetArabicProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=arabic

[DetectArabicISO]
Property=SetArabicISOProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=arabicISO,ISOarab*

[DetectEnglish]
Property=SetEnglishProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*eng*,uk,*british

[DetectChSimplified]
Property=SetChSimplifiedProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*ChSimp*,ChineseSimp*

[DetectChTraditonal]
Property=SetChTraditionalProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE,*/LANG
PropertyMatch=*ChTrad*,ChineseTrad*

[DetectFrench]
Property=SetFrenchProperty
PropertyFieldCSVs=*/DRELANGUAGETYPE, */DRELANGAGETYPE,*/LANG
PropertyMatch=*fre*,fran*
```

For each Property that you have defined in the FieldProcessing sections, you need to define a section with the same value of the respective property. In this section you can then specify the language type (which you also need to list in the DRE's **[LanguageTypes]** section where you define how you want the DRE to handle the individual languages).

For example:

```
[SetArabicProperty]
LanguageType=Arabic
HiddenType=TRUE

[SetArabicISOProperty]
LanguageType=ArabicISO
HiddenType=TRUE

[SetEnglishProperty]
LanguageType=English
HiddenType=TRUE

[SetChSimplifiedProperty]
LanguageType=ChSimplified
HiddenType=TRUE

[SetChTraditionalProperty]
LanguageType=ChTraditional
HiddenType=TRUE

[SetFrenchProperty]
LanguageType=French
HiddenType=TRUE
```

Note: you can override these values at index and query time by using the **LanguageType** parameter in the command.

Defining language types in the DRE configuration file

In order to run the DRE in multiple languages, you need to specify language types for each of the language and encoding combination that you want to use in the **[LanguageTypes]** section of the DRE's configuration file.

For example:

```
[LanguageTypes]
0=arabic
1=arabicISO
2=english
3=chSimplified
4=chTraditional
5=french
```

For each of the language types that you have defined in the **[LanguageTypes]** section, you need to define a section with the same name. In this section you can then specify appropriate settings that determine how the DRE handles this language type (for example, the name of the sentence breaking library module, if required).

Note:

- in DRE 4 the **StripLanguage** and **CharConv** settings have been deprecated (the functionality has been automated according to language and encoding).
- for details on the configuration settings that you can use for the languages in the **[LanguageTypes]** section, please refer to the DRE 4 configuration help HTML file, which is located in your DRE 4 directory.

For example:

```
[arabic]
LanguageCode=1
Language=ARABIC
Encoding=ARABIC

[arabicISO]
LanguageCode=2
Language=ARABIC
Encoding=ARABIC_ISO
```

```
[english]
LanguageCode=3
Language=ENGLISH
Encoding=ASCII

[chSimplified]
LanguageCode=4
SentenceBreaking=chinesebreaking.dll
Language=CHINESE
Encoding=CHINESESIMPLIFIED

[chTraditional]
LanguageCode=5
SentenceBreaking=chinesebreaking.dll
Language=CHINESE
Encoding=CHINESETRADITIONAL

[french]
LanguageCode=6
Language=FRENCH
Encoding=ASCII
```

Tip:

If the field in that indicates the language of a document is the same in all the documents that your DRE is handling, you can configure your DRE as follows:

1. Set up a process for looking up the language of a document in the [FieldProcessing] section.

   For example:

   ```
   [FieldProcessing]
   Number=1
   0=LookForLanguage
   ```

2. Create a section for the process, in which you create a Property for the process and identify the field that you want to apply the process to.

   For example:

   ```
   [LookForLanguage]
   Property=SetLanguage
   PropertyFieldCSVs=*/DRELANGUAGE
   ```

3. List the Property that you have created in the [Properties] section.

   For example:

   ```
   [Properties]
   0=SetLanguage
   ```

4. Create a section for this property, in which you set the **LanguageType** parameter to **true** to map the values of the */DRELANGUAGE fields to the equivalent language type in the [LanguageTypes] section.

   For example:

   ```
   [SetLanguage]
   LanguageType=true

   [LanguageTypes]
   0=russianISO
   1=russianKOI8
   2=russianUTF8

   [russianISO]
   LanguageCode=1
   Language=Russian
   Encoding=CYRILLIC_ISO

   [russianKOI8]
   LanguageCode=2
   Language=Russian
   Encoding=CYRILLIC_KOI8

   [russianUTF8]
   LanguageCode=3
   Language=Russian
   Encoding=UTF8
   ```

Encoding settings for supported languages

Note:

- All DRE **Encoding** settings can alternatively be set to UTF8 or UCS2.
- The internal DRE storage encoding is UTF8.

Afrikaans

Character set:	Latin1
Set DRE Language parameter to:	AFRIKAANS
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Albanian

Character set:	Latin1
Set DRE Language parameter to:	ALBANIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Arabic

Character set:	Cyrillic
Set DRE Language parameter to:	ARABIC
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1256	ARABIC
iso-8859-6	ARABIC_ISO

Azeri

Character set:	Cyrillic
Set DRE Language parameter to:	AZERI
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Basque

Character set:	Latin1
Set DRE Language parameter to:	BASQUE
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Belarussian

Character set:	Cyrillic
Set DRE Language parameter to:	BELARUSSIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Bulgarian

Character set:	Cyrillic
Set DRE Language parameter to:	BULGARIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Catalan

Character set:	Latin1
Set DRE Language parameter to:	CATALAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Chinese Traditional

Character set:	Big-5
Set DRE Language parameter to:	CHINESE
For Encoding:	**set DRE Encoding parameter to:**
Big-5	CHINESETRADITIONAL

Chinese Simplified

Character set:	GB2312-80
Set DRE Language parameter to:	CHINESE
For Encoding:	**set DRE Encoding parameter to:**
windows-CP950	CHINESESIMPLIFIED

Croatian

Character set:	Latin2
Set DRE Language parameter to:	CROATIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Czech

Character set:	Latin2
Set DRE Language parameter to:	CZECH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Danish*

Character set:	Latin1
Set DRE Language parameter to:	DANISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Dutch*

Character set:	Latin1
Set DRE Language parameter to:	DUTCH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

English*

Character set:	Latin1
Set DRE Language parameter to:	ENGLISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Estonian

Character set:	Latin4
Set DRE Language parameter to:	ESTONIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1257	NORTHERNEUROPEAN
iso-8859-4	NORTHERNEUROPEAN_ISO

Faroese

Character set:	Latin1
Set DRE Language parameter to:	FAROESE
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

Finnish

Character set:	Latin1
Set DRE Language parameter to:	FINNISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

French*

Character set:	Latin1
Set DRE Language parameter to:	FRENCH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Galician *

Character set:	Latin1
Set DRE Language parameter to:	GALICIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

German*

Character set:	Latin1
Set DRE Language parameter to:	GERMAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Greek*

Character set:	Greek
Set DRE Language parameter to:	GREEK
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1253	GREEK
iso-8859-7	GREEK_ISO

Greenlandic

Character set:	Latin4
Set DRE Language parameter to:	GREENLANDIC
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1257	NORTHERNEUROPEAN
iso-8859-4	NORTHERNEUROPEAN_ISO

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

Hebrew

Character set:	Hebrew
Set DRE Language parameter to:	HEBREW
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1255	HEBREW
iso-8859-8	HEBREW_ISO

Hungarian

Character set:	Latin2
Set DRE Language parameter to:	HUNGARIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Icelandic

Character set:	Latin1
Set DRE Language parameter to:	ICELANDIC
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Indonesian

Character set:	Latin1
Set DRE Language parameter to:	INDONESIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Italian*

Character set:	Latin1
Set DRE Language parameter to:	ITALIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Japanese**

Character set:	Japanese
Set DRE Language parameter to:	JAPANESE
For Encoding:	**set DRE Encoding parameter to:**
Shift-JIS	SHIFTJIS
EUC	EUC
JIS	JIS

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

****** the language has stemming embedded in sentence breaking.

Kazakh

Character set:	Cyrillic
Set DRE Language parameter to:	KAZAKH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Korean**

Character set:	Korean
Set DRE Language parameter to:	KOREAN
For Encoding:	**set DRE Encoding parameter to:**
KS C 5601-1987	KOREAN
KS C 5601-1992	KOREAN

Kyrgyz

Character set:	Cyrillic
Set DRE Language parameter to:	KYRGYZ
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

****** the language has stemming embedded in sentence breaking.

Lappish

Character set:	Latin4
Set DRE Language parameter to:	LAPPISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1257	NORTHERNEUROPEAN
iso-8859-4	NORTHERNEUROPEAN_ISO

Latvian

Character set:	Latin4
Set DRE Language parameter to:	LATVIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1257	NORTHERNEUROPEAN
iso-8859-4	NORTHERNEUROPEAN_ISO

Lithuanian

Character set:	Latin4
Set DRE Language parameter to:	LITHUANIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1257	NORTHERNEUROPEAN
iso-8859-4	NORTHERNEUROPEAN_ISO

Macedonian

Character set:	Cyrillic
Set DRE Language parameter to:	MACEDONIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Malay

Character set:	Latin1
Set DRE Language parameter to:	MALAY
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Maori

Character set:	Latin1
Set DRE Language parameter to:	MAORI
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Mongolian

Character set:	Cyrillic
Set DRE Language parameter to:	MONGOLIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Norwegian*

Character set:	Latin1
Set DRE Language parameter to:	NORWEGIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Polish

Character set:	Latin2
Set DRE Language parameter to:	POLISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Portuguese*

Character set:	Latin1
Set DRE Language parameter to:	PORTUGUESE
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Romanian

Character set:	Latin2
Set DRE Language parameter to:	ROMANIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Russian*

Character set:	Cyrillic
Set DRE Language parameter to:	RUSSIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

Serbian

Character set:	Cyrillic
Set DRE Language parameter to:	SERBIAN

For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Slovak

Character set:	Latin2
Set DRE Language parameter to:	SLOVAK

For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Slovenian

Character set:	Latin2
Set DRE Language parameter to:	SLOVENIAN

For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Sorbian

Character set:	Latin2
Set DRE Language parameter to:	SORBIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1250	EASTERNEUROPEAN
iso-8859-2	EASTERNEUROPEAN_ISO

Spanish*

Character set:	Latin1
Set DRE Language parameter to:	SPANISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Swahili

Character set:	Latin1
Set DRE Language parameter to:	SWAHILI
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

Swedish*

Character set:	Latin1
Set DRE Language parameter to:	SWEDISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

Tatar

Character set:	Cyrillic
Set DRE Language parameter to:	TATAR
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Thai

Character set:	Thai
Set DRE Language parameter to:	THAI
For Encoding:	**set DRE Encoding parameter to:**
windows-CP874 / iso-8859-11	THAI

***** the language has stemming implemented. If you set **Stemming** to **true** for a language in the configuration and no stemming algorithm is available, the setting will have no effect.

Turkish

Character set:	Latin5
Set DRE Language parameter to:	TURKISH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1254 / iso-8859-9	TURKISH

Ukrainian

Character set:	Cyrillic
Set DRE Language parameter to:	UKRAINIAN
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Uzbek

Character set:	Cyrillic
Set DRE Language parameter to:	UZBEK
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1251	CYRILLIC
KOI8-R	CYRILLIC_KOI8
iso-8859-5	CYRILLIC_ISO

Vietnamese

Character set:	Vietnamese
Set DRE Language parameter to:	VIETNAMESE
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1258	VIETNAMESE
VISCII	VIETNAMESE_VISCII

Welsh

Character set:	Latin1
Set DRE Language parameter to:	WELSH
For Encoding:	**set DRE Encoding parameter to:**
windows-CP1252 / iso-8859-1	ASCII

TermSize setting for supported languages

Allows you to specify the maximum number of characters that any term in the DRE can be. By default this is 10 (for English and other European languages). The recommended values for the different languages are:

20	English and other European languages	**60**	German
30	Korean	**30**	Thai
30	Japanese		

Transliteration settings for supported languages

Transliteration is the process of converting accented characters such as øù into equivalent non accented characters. This is useful in environments where accented keyboards are not available.

Transliteration is always applied to the following languages:

Language:	Transliteration:
Japanese	Half width katakana to full width katakana Full width 0-9, A-Z, a-z to single byte 0-9, A-Z, a-z
Chinese	Full width 0-9, A-Z, a-z to single byte 0-9, A-Z, a-z
Greek	Accented Greek characters to non accented characters
Spanish	Accented vowels áéíóúü to non accented vowels
Portuguese	Accented vowels àáâãçéêíòóôõúü to non accented vowels

Transliteration is optional for the following languages:

Language	Transliteration		
Western European	àáâãä=a	å=aa	ç=c
	èéêë=e	ìíîï=i	òóôõö=o
	ø=oe	ùúûü=u	œ(oe)=oe
	æ=ae	ß=ss	ñ=nh
	ý=y	ð=d	þ=th
German	Same as Western European apart from: ä=ae, ö=oe, ü=ue		
Scandinavian	Same as Western European apart from: ä=ae, ö=oe, ü=ue		
Russian	All characters mapped to A-Z		

SentenceBreaking files for supported languages

If you want to use languages in which words are not delimited by spaces (Japanese, Chinese, Thai and Korean), you need to place external sentence breaking files into the directory that contains the DRE. If you are running your DRE on a UNIX platform, you need to specify the LD_LIBRARY_PATH to ensure that the DRE can find the sentence breaking files that it requires.

The following table lists the files that the individual languages require and the language settings that you need to enter in the DRE's configuration file.

Japanese

Required files:	**NT**	**UNIX**
	japanesebreaking.dll	japanesebreaking.so
	\dic\jtag.attr	/dic/system/jtag.attr
	dic\JTAG.hash	/dic/system/jtag.hash
	dic\jtag.id	/dic/system/jtag.id
	\dic\jtag.mrph	/dic/system/jtag.mrph
	dic\JTAG.offset	/dic/system/jtag.offset
	\dic\jtag.table	/dic/system/jtag.table
	jtag.dll	/dic/system/jtag.trie
	jtag.ini	jtag.ini
	jtag_at.dll	libcodeconv.so
	dic\JTAG.trie	

Traditional Chinese

Required files:	**NT**	**UNIX**
	chinesebreaking.dll	chinesebreaking.so
	big5togb.txt	big5togb.txt
	wordlist.txt	wordlist.txt

Simplified Chinese

Required files:	NT	UNIX
	chinesebreaking.dll	chinesebreaking.so
	big5togb.txt	big5togb.txt
	wordlist.txt	wordlist.txt

Thai

Required files:	NT	UNIX
	thaibreaking.dll	thaibreaking.so
	thaidict.txt	thaidict.txt

Korean

Required files:	NT	UNIX
	koreanbreaking.dll	koreanbreaking.so
	Koma.dll (NT only)	main.dat
	HanTag.dll (NT only)	prob.dat
	main.dat	main.fst
	prob.dat	prob.fst
	main.fst	pos.nam
	prob.fst	tag.nam
	pos.nam	tagout.nam
	tag.nam	connection.txt
	tagout.nam	stopposnam.txt
	connection.txt	tagname.txt
	stopposnam.txt	
	tagname.txt	

Note: by default all words are stemmed to verbs. If you want to stem to nouns rather than verbs, you need to set the following sentence breaking settings in the **stopposnam.txt** file to **0** (by default they are set to **1**).

Stoplists for supported languages

Each language that is supported needs a stoplist (if a stop list isn't provided for a language, you can create one), which contains a list of common words that are not indexed into the DRE. Words as, for example, "the" or "a" are used too frequently to carry any significance and the DRE does not require them to understand the concept of text.

You can use a standard text editor to edit the stoplist that your DRE uses, for example, if you want to add other words that are common to most or all of your documents.

You can list the words in the stoplist in any of the valid encodings for that language (for example, in Russian you could specify stopwords in KOI8, UTF8, ISO and so on). You can use different encodings within the same stoplist file (see **Encoding settings for supported languages**).

You only need to specify each word once. There is, for example, no need to specify a word in several different encodings.

For all operations, the DRE will recognize words as stopwords irrespective of the encoding they are given in; for example, in Russian you could list a stopword in the KOI8 encoding in the stoplist file and it would be recognized if it occurred in a document in UTF8.

Note:

For each encoding that you want to use you must create a section in your stoplist file. Name the section after the language type that you are using (the language types are listed in the "**Set DRE Encoding parameter to**" column of the "**Encoding settings for supported languages**" list). Words can be in upper or lower case, and can be separated by spaces or new lines.

For example:

```
[cyrillic_koi8]
äáöå
äìñ äï
åå åçï

[cyrillic_iso]
° ±µ· ±¾»µµ ±Ë ±Ë»s
```

In this example, a Russian stoplist contains 10 words, of which 5 are in CYRILLIC_KOI8 encoding and five are in the CYRILLIC_ISO encoding.

Indexing multi-lingual documents

If your DRE license enables Automatic Language Detection and you have set **AutoDetectLanguagesAtIndex** to **true** in the DRE's configuration file, the DRE automatically identifies the language and encoding of a document when it is indexed.

If your license does not include this functionality, the DRE assumes that all documents that are indexed have the specified **DefaultLanguageType**, unless you do one of the following:

- Specify a field process in the DRE configuration file's **[FieldProcessing]** section, which allow the DRE to read the language of a document from one of its fields (see **Configuring a multi-lingual DRE**).

- Specify the language and encoding of documents that you are indexing into the DRE.

  If you have set up a field process to identify a document's language and encoding from one of its fields, you can also do this to overwrite the language and encoding that the DRE would normally read from the document's field.

  To specify the language type (the language and the encoding for this language) of a document you are indexing, you need to add the **LanguageType** parameter to your indexing command (you define the available language types in the **[LanguageTypes]** section of your DRE configuration file).

  Examples:

  http://12.3.4.56:4001/DREADD?MyIndexFile.xml&LanguageType=Japanese_SHIFTJIS

  In this example, the **MyIndexFile.xml** file is indexed with the language type **Japanese_SHIFTJIS**. The way the DRE handles this language type is determined by the way it has been defined in the DRE's configuration file (that is by the settings that you have associated with this language type in the configuration file).

  http://12.3.4.56:4001/DREADD?C:\autonomy\autoindexer\myjob.idx&LanguageType=RussianKOI8

  In this example, the **myjob.idx** file is indexed with the language type **RussianKOI8**. The way the DRE handles this language type is determined by the way it has been defined in the DRE's configuration file (that is by the settings that you have associated with this language type in the configuration file).

Note:

For further details on how to index files into the DRE, please refer to the **Importing and indexing data into the DRE** chapter.

Specifying the language type of your query

When you send a query that includes explicit query text to the DRE, it assumes by default that this query is in the language and encoding specified for the **DefaultLanguageType** in the DRE configuration file. It is essential that the DRE knows the language type of any query text that is submitted to it, so that can handle it appropriately (for example, by applying correct stemming and so on).

If you want to send a query that uses another language and/or encoding, you need to add the **LanguageType** parameter to your query command, which instructs the DRE that this query uses the language and encoding that have been set in the DRE configuration file for the specified **LanguageType**.

For example:

The following language types are specified in the DRE configuration file:

```
[LanguageTypes]
DefaultLanguageType=English
LanguageDirectory=./langfiles
0=English
1=Deutsch
Encoding=ASCII

[English]
LanguageCode=1
Language=ENGLISH
Encoding=ASCII

[Deutsch]
LanguageCode=2
Language=GERMAN
Encoding=ASCII
```

The following query uses the language and encoding that has been specified for the **DefaultLanguageType**, so you can send it to the DRE without adding the **LanguageType** parameter:

> **http://12.3.4.56:4000/action=Query&Text=The Bayes theory of probability**

The following query uses the language and encoding that has been specified for the **Deutsch** language type. Therefore you must add the **LanguageType** parameter to the query, so the DRE can determine the language and encoding it uses:

> **http://12.3.4.56:4000/action=Query&Text=Einsteins Relativitätstheorie&LanguageType=Deutsch**

Note:

For further details on how to send query commands to the DRE, please refer to the **Querying the DRE** chapter.

Converting documents to a specific encoding when querying the DRE

You can send the following types of query to the DRE:

- **Text queries**

  Queries that contain some form of query text (for example, Query, SuggestOnText, Summarize and so on).

- **Text-free queries**

  Queries that do not contains any query text (for example, Suggest, List, GetContent and so on).

Text queries

When you send a query action to the DRE, it returns by default results that use the same language and encoding as the query text (that is the language that has been specified for the **LanguageType** that is sent with the query, or for the **DefaultLanguageType** if no **LanguageType** is sent with the query).

If you want a query action to return results in a specific encoding, you must add the **OutputEncoding** to your query. This parameter allows you to convert the results of a query to any type of encoding that is compatible with the query's language (if you specify an encoding that is not compatible with the query's language, the DRE indicates this in the results).

Text-free queries

Query actions that do not contain any query text by default return results in the **OutputEncoding** that has been specified for the **DefaultLanguageType**. If any of the query's results is not compatible with this encoding, the DRE indicates this in the results.

If you want a query action to return results in a specific encoding, you can add the **OutputEncoding** to your query. The DRE to converts all results to this encoding, provided they are compatible with it. If any of the query's results is not compatible with this encoding the DRE returns an appropriate message.

For example:

http://12.3.4.56:4000/action=Query&Text=Neurologia i Neurochirurgia&LanguageType=PolishEASTERNEUROPEAN&OutputEncoding=EASTERNEUROPEAN_ISO

http://12.3.4.56:4000/action=Suggest&ID=9016&OutputEncoding=EASTERNEUROPEAN_ISO

In these examples, the DRE will convert all query results to **EASTERNEUROPEAN_ISO**.

Returning documents in multiple languages for your query

When you send a query to the DRE, it returns by default results that use the same language as the query text (that is the language that has been specified for the **LanguageType** that is sent with the query, or for the **DefaultLanguageType** if no **LanguageType** is sent with the query).

If you want to return documents in any language for your query rather than only in the query's language, you need to add **AnyLanguage=true** to your query.

Note that the query will only return documents in multiple languages if they contain terms that match terms in the query (for example, query text that contains the term "marketing" might return documents in English, French, German and so on).

3. Administering the DRE

You can administer the DRE using:

- **Index port commands**

  You can send index port commands to your DRE using your web browser or by writing your own script (note that you cannot send some commands via your web browser as they require POST request methods). Alternatively, you can use the DRE API (for example C, Java, VB, JNI and so on) in order to write code that allows you to administer your DRE.

  You can use index port commands to:

  - execute DRE configuration changes
  - delete documents from the DRE by reference
  - delete individual documents and ranges of documents from the DRE
  - create new DRE databases
  - delete a database and its documents
  - delete all documents from a database
  - expire documents
  - compact the DRE
  - change field values in DRE documents
  - back up the DRE
  - initialize the DRE

- **the DRE Administration dialog**

  If you are running on a Windows NT or 2000 platform, you can use the DRE Administration dialog to administer the DRE. Please refer to **Appendix C** for further details on the DRE Administration dialog.

Executing configuration changes

If you have made changes to the DRE's configuration file, you need to do one of the following to ensure that the DRE acknowledges them:

- send a DRERESET command
- stop and restart the DRE

To send a DRERESET command to the DRE

Issue the following command from your web browser:

http://<host>:<port>/DRERESET?

<host>

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

<port>

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

For example:

http://12.3.4.56:4001/DRERESET?

This command uses port **4001** to reset the DRE that is located on a machine with the IP address **12.3.4.56**.

To stop and restart the DRE

For Windows:

Display the Windows Services dialog, stop the DRE service and then start it again.

For UNIX:

Use the **Stop.sh** stop script to stop the DRE and then start it again using the **Start.sh** script (the scripts are supplied in the DRE installer).

Deleting documents from the DRE by reference

You can delete one or more documents from the DRE by reference by issuing the following command from your web server:

http://<host>:<port>/DREDELETEREF?docs=<document_references>&field=<fields>&DREdbname=<database_name>

<host>

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

<port>

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

<document_references>

Enter the references of the documents that you want to delete. If you want to specify multiple references, you must separate them with plus symbols (there must be no space before or after a plus symbol).

<fields>

This parameter is optional.

Allows you to restrict which documents are deleted by specifying one or more fields that a document must contain in order to be deleted. Only documents that have one of the specified references and at least one of the specified fields are deleted.

If you want to specify multiple fields, you must separate them with commas or spaces (there must be no space before or after a comma).

<database_name>

This parameter is optional.

Allows you to specify the name of the database, which contains the documents that you want to delete. If you don't specify a database and the specified document is contained in several databases, it is deleted from all of them.

For example:

http://12.3.4.56:4001/DREDELETEREF?docs=http%3A%2F%2Fnews%2Enewssite%2Ecom%2Findex%2Ehtml+http%3A%2F%2Fnews%2Enewssite%2Ecom%2Fcoverstory%2Ehtml

This command uses port **4001** to delete the documents with the specified URLs from the DRE that is located on a machine with the IP address **12.3.4.56**.

Deleting individual documents and ranges of documents from the DRE

You can delete individual documents and / or ranges of documents from the DRE by issuing the following command from your web server:

http://<host>:<port>/DREDELETEDOC?docs=<document_references>&range=[<doc_ref>,<doc_ref>]

The **DREDELETEDOC?** command allows you delete individual documents and / or ranges of documents from the DRE by their DOCID.

<host>

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

<port>

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

<document_references>

Enter the DOCID of the document that you want to delete. If you want to specify multiple references, you must separate them with plus symbols (there must be no space before or after a plus symbol).

<doc_ref>,<doc_ref>

Enter the DOCID of the first and last document in the range of documents that you want to delete.

For example:

http://12.3.4.56:4001/DREDELETEDOC?docs=3+5+range=[7-10]

This command uses port **4001** to delete the documents with the DOCID **3**, **5**, **7**,**8**,**9** and **10** from the DRE that is located on a machine with the IP address **12.3.4.56**.

Creating a new database in the DRE

You can create a new database in the DRE (for example, to store documents that related to one particular subject or to store documents that are relevant to a particular user group), by doing one of the following:

- send a DRECREATEDBASE command
- add the database to the DRE configuration file

To send a DRECREATEDBASE command to the DRE

Issue the following command from your web browser:

```
http://<host>:<port>/DRECREATEDBASE?DREdbname=<database_name>
```

<host>

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

<port>

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

<database_name>

Enter the name of the database that you want to create in the DRE.

For example:

```
http://12.3.4.56:4001/DRECREATEDBASE?DREdbname=Archive
```

This command uses port **4001** to create a new **Archive** database in the DRE that is located on a machine with the IP address **12.3.4.56**.

To add a database to the DRE configuration file

1. Open the DRE configuration file in a text editor and find the [Databases] section.
2. Add a new database to the list of databases that the [Databases] section contains and increase the **NumDBs** setting by one.

   For example:

   ```
   [Databases]
   NumDBs=2
   0=News
   1=Archive
   ```

   The above [Databases] section lists two DRE databases. To add a new database, you need to add a new line to the list:

   ```
   [Databases]
   NumDBs=3
   0=News
   1=Archive
   2=MyNewDatabase
   ```

   Note that the listed databases are numbered in consecutive order, starting from **0**.
3. Save and close the configuration file.
4. Restart the DRE to execute your changes.

Deleting a database and all the documents it contains

You can delete a DRE database and all the documents it contains by issuing the following command from your web server:

http://<host>:<port>/DREREMOVEDBASE?DREdbname=<database_name>

<host>

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

<port>

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

<database_name>

Enter the name of the database that you want to delete from the DRE. The documents in this database are deleted from the DRE as well.

For example:

http://12.3.4.56:4001/DREREMOVEDBASE?DREdbname=Archive

This command uses port **4001** to delete the **Archive** database and all documents that this database contains from the DRE that is located on a machine with the IP address **12.3.4.56**.

Deleting all documents from a database

You can delete all documents from a DRE database by issuing the following command from your web server:

http://<host>:<port>/DREDELDBASE?DREdbname=<database_name>

<host>

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

<port>

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

<database_name>

Enter the name of the database from which you want to delete all documents.

Example:

http://12.3.4.56:4001/DREDELDBASE?DREdbname=Archive

This command uses port **4001** to delete all documents from the DRE's **Archive** database. The DRE is located on a machine with the IP address **12.3.4.56**.

Expiring documents

In order to ensure that the documents in your DRE are up to date, you can execute an Expiry operation which deletes or archives documents that have reached a specific age. You can expire documents:

- immediately by sending a DREEXPIRE command
- in regular intervals by setting up a scheduled Expiry operation

By default documents are deleted when they expire. If you want to archive them instead, enter the name of the database that you want to use for archiving for the **ExpireIntoDatabase** setting in the DRE configuration file's [Schedule] section.

The date that determines whether a document should be expired can be read from a field in the document or from the expiry time that has been set for the database that contains the document. If the DRE is unable to determine whether a document should be expired (because the document does not contain a field that sets its expiry date and the document's database has no expiry time set) the DRE does not expire the document.

To set up fields that determine when documents should expire

1. Open the DRE configuration file in a text editor and find the [FieldProcessing] section.
2. Add a new field process to the list of field processes that the [FieldProcessing] section contains and increase the **Number** setting by one.

   For example:

   ```
   [FieldProcesses]
   Number=2
   0=IndexFields
   1=IndexAndWeightHigher
   ```

   The above [FieldProcessing] section lists two field processes. To add a new field process, you need to add a new line to the list:

   ```
   [FieldProcesses]
   Number=3
   0=IndexFields
   1=IndexAndWeightHigher
   2=ExpireDateFields
   ```

   Note that the listed field processes are numbered in consecutive order, starting from **0**.
3. Create a section for your new field process in the configuration file. Create a property for the new process and use the **PropertyFieldCSVs** settings to identify the document fields that should determine whether documents should be expired (a document expires once the time in this field has elapsed).

For example:

```
[ExpireDateFields]
Property=SetExpireDate
PropertyFieldCSVs=*/DREEXPIRE,*/valid_time
```

4. Find the [Properties] section and add your new property to the list of properties that the [Properties] section contains.

   For example:

```
[Properties]
0=Index
1=IndexWeight
3=SetExpireDate
```

5. Create a section for your new property in the configuration file and set the **ExpireDateType** to **true** in order to indicate that the associated **PropertyFieldCSVs** fields hold the document expiry date.

   For example:

```
[SetExpireDate]
ExpireDateType=TRUE
```

To expire documents immediately

Issue the following command from your web browser:

http://<host>:<port>/DREEXPIRE

<host>

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

<port>

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

For example:

http://12.3.4.56:4001/DREEXPIRE

This command uses port **4001** to expire the documents from the DRE that is located on a machine with the IP address **12.3.4.56**.

To expire documents in regular intervals

1. Open the DRE configuration file in a text editor and find the [Schedule] section. If the configuration file doe not contain a [Schedule] section, you can add one.
2. Specify the following settings within the [Schedule] section:

   Expire

   Enter **true** to enable an Expiry schedule.

   ExpireTime

   Specify the time (hh:mm) when you want the Expiry operation to start.

   ExpireInterval

   Specify the number of hours that elapse between individual Expiry operations. Enter **0** if you want the operation to take place daily.

   ExpireIntoDatabase

   Enter the name of the database that you want to use to archive expired documents. If you want documents to be deleted when they expire, don't specify this setting.

   For example:

   ```
   [Schedule]
   Expire=true
   ExpireTime=00:00
   ExpireInterval=24
   ExpireIntodatabase=Archive
   ```

Compacting the DRE

You can reduce the space that documents take up by executing a Compact operation. This operation fills up the space that has been created through the deletion of documents with new documents (similar to the defragmentation process).

You can compact the DRE

- immediately by sending a DRECREATEDBASE command
- in regular intervals by setting up a scheduled Compact operation

To compact the DRE immediately

Issue the following command from your web browser:

http://<host>:<port>/DRECOMPACT

<host>

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

<port>

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

For example:

http://12.3.4.56:4001/DRECOMPACT

This command uses port **4001** to compact the DRE that is located on a machine with the IP address **12.3.4.56**.

To compact the DRE in regular intervals

1. Open the DRE configuration file in a text editor and find the [Schedule] section. If the configuration file doe not contain a [Schedule] section, you can add one.
2. Specify the following settings within the [Schedule] section:

   Compact

   Enter **true** to enable a Compact schedule.

   CompactTime

   Specify the time (hh:mm) when you want the Compact operation to start.

   CompactInterval

   Specify the number of hours that elapse between individual Compact operations. Enter **0** if you want the operation to take place daily.

   For example:

   ```
   [Schedule]
   Compact=true
   CompactTime=00:00
   CompactInterval=24
   ```

Changing field values in DRE documents

You can change the values of fields in documents that have been indexed into the DRE using the following command:

DREREPLACE?<data>#DREENDDATA

Note: This command requires a POST request method

<data>

The fields that you want to replace in the DRE. You need to specify each field as follows:

#DREDOCID ***n*** or #DREDOCREF ***n***

#DREFIELDNAME ***x***

#DREFIELDVALUE ***y***

n

The DocID or reference (URL) of the document that contains the field, which you want to replace.

x

The name of the field whose value you want to change.

y

The value that you want field x to change to. For example:

```
#DREDOCID 1
#DREFIELDNAME Price
#DREFIELDVALUE 10
#DREDOCREF http://www.autonomy.com/autonomy/dynamic/autopage442.shtml
#DREFIELDNAME Country
#DREFIELDVALUE UK
#DREENDDATA
```

In this example, the value of the **Price** field in the document with the DocID **1** is changed to **10**. The value of the **Country** field in the document with the reference **http://www.autonomy.com/autonomy/dynamic/autopage442.shtml** is changed to **UK**.

Backing up the DRE

In order to create a safe copy of the DRE, you can back up your DRE.

You can back up the DRE

- immediately by sending a DREBACKUP command
- in regular intervals by setting up a scheduled backup

To back up your DRE immediately:

1. Send a Compact command to the DRE in order to compress it (see **Compacting the DRE**).
2. Issue the following command from your web browser to copy all the DRE's *.DB files to an new location:

   http://<host>:<port>/DREBACKUP?<path>

   <host>

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

   <port>

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

   <path>

   Enter the path to the location where you want to create the DRE backup.

   For example:

   http://12.3.4.56:4001/DREBACKUP?E:\DRE_Backup

   This command uses port **4001** to create a backup of the DRE that is located on a machine with the IP address **12.3.4.56** on **E:\DRE_Backup**.
3. Issue the following command from your web browser in order to restore the files to a DRE:

   http://<host>:<port>/DREINITIAL?<path>

   <host>

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

   <port>

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

<path>

Enter the path to the location of the DRE backup.

For example:

http://12.3.4.56:4001/DREINITIAL?E:\DRE_Backup

This command uses port **4001** to restore the files backed up on **E:\DRE_Backup** to the DRE that is located on a machine with the IP address **12.3.4.56**.

To back up your DRE in regular intervals:

1. Open the DRE configuration file in a text editor and find the [Schedule] section. If the configuration file doe not contain a [Schedule] section, you can add one.
2. Specify the following settings within the [Schedule] section:

   Backup

   Enter **true** to enable a schedule backup.

   BackupCompression

   Enter **true** to compress the DRE before it is backed up.

   BackupTime

   Specify the time (hh:mm) when you want the backup to start.

   BackupInterval

   Specify the number of hours that elapse between individual backups. Enter **0** if you want the backup to take place daily.

   NumberOfBackups

   Specify the number of times you want to back up the DRE. You can cycle the backing up procedure by specifying multiple backups (note that the number of backups you specify must correspond to the number of **BackupDirN** directories you specify). Multiple backups are executed as follows:

   The first backup is created at the specified **BackupTime**, the next one is created after the specified **BackupInterval** and so on. Once the DRE has been backed up as many times as you have specified for **NumberOfBackups**, the first backup is overwritten the next time the specified **BackupInterval** elapses. This means that you always have an up to date set of DRE backups.

BackupDirN

Enter the path to the location where you want to create the DRE backup. You must specify one directory for each of the **NumberOfBackups**.

BackupMaintainDirStructure

Enter **true** to maintain the directory structure when the DRE is backed up.

For example:

```
[Schedule]
Backup=true
BackupCompression=true
BackupTime=00:00
BackupInterval=24
BackupMaintainStructure=true
NumberOfBackups=3
BackupDir0=E:\DRE_Backup0
BackupDir1=E:\DRE_Backup1
BackupDir2=E:\DRE_Backup2
```

Initializing the DRE

You can get rid of the data that your DRE contains and reset it to the state it was in when you first installed it. Issue the following command from your web browser to reset the DRE:

http://<host>:<port>/DREINITIAL?

<host>

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

<port>

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

For example:

http://12.3.4.56:4001/DREINITIAL?

This command uses port **4001** to reset the DRE that is located on a machine with the IP address **12.3.4.56** to its original state.

Appendix A: upgrading from an earlier DRE version

DRE4 cannot read DRE3.X or earlier databases. This means that if you are migrating from an earlier DRE, you must re-index all data that your old DRE contains (this is only possible if you have stored the plain text content in the DRE).

To re-index the DRE content:

Enter the following command in the address field of your web browser:

HTTP://**<Host>**:**<QueryPort>/**qmethod=G&xoptions=nocombine+outfile=**<AllDocs>**.idx

<Host>

The IP address (or name) of the machine on which your old DRE is running.

<QueryPort>

The port number by which queries are sent to the DRE.

<AllDocs>

The name of the file in which you want the content of your old DRE to be stored. This file has to be in IDX file format.

An IDX file is created, which contains the DRE's plain text content.

Note:

Before you index this file into your DRE 4, you need to set up databases that match the databases of your old DRE. You can do this by adding the databases to the [Databases] section in your DRE 4 / AXE's configuration file.

Upgrading within an Autonomy environment

If you are upgrading a DRE which is part of an environment that comprises other Autonomy solutions (for example, Connectors, a Distributed Query Handler, a Distributed Index Handler, Portal-in-a-Box and so on), and you install your DRE 4 / AXE in the location that was previously used by your old DRE, the components will continue to communicate with each other.

However, if your new DRE is located on a different machine or uses different ports, you need to reflect this by making appropriate changes to the configuration settings of the Autonomy solutions that you want to use to your new DRE. You need to change any instance of the following settings:

DRE*n*Host

DRE*n*Port

Engine*n*

DREHost

IndexPort

QueryPort

Appendix B: DRE 3 backwards compatibility

DRE 4 / AXE supports all DRE 3 qmethods, which allows you to integrate with applications that are not ACI compatible (for example, Portal-in-a-Box 3).

The following has changed in DRE 4 / AXE:

- Specifying databases for indexing
- Processing fields
- Using Security modules
- Using templates
- Using a Thesaurus DRE

Note: if you are upgrading from DRE 3, you must re-index your data into to DRE 4 / AXE.

Specifying databases for indexing

In DRE 3 only the DREDBNAME field could contain the name of the database into which a document should be indexed.

In DRE 4 you can specify the fields from which you want the DRE to read into which database it should index a file.

For example:

```
[DatabaseFields]
Property=Database
PropertyFieldCSVs=*/DREDBNAME,*/DB,*/Database

[Properties]
0=Database

[Database]
DatabaseType=TRUE
```

Processing fields

In DRE 3 you were required to specify each field that you wanted to index, apply a higher weighting to or use as a reference.

In DRE 4 / AXE all fields that documents contain are automatically stored in the DRE. You can identify fields using wildcards and apply any type of processing to them or the documents that contain them. You can also make the processing of fields dependent on the value that the fields contain.

For further details on field processing in DRE 4 please refer to the chapter **Configuring the DRE**.

Example

In DRE 3:

```
[Fields]
NumFields=6
0=DREREFERENCE
1=INDEX10.DRETITLE
2=REF.200,URL
3=INDEX.TITLE
4=INDEX.SUMMARY
5=INDEX.AUTHOR
```

Equivalent in DRE 4:

```
[FieldProcessing]
Number=3
0=SetReferenceFields
1= IndexFields
2= IndexAndWeightHigher

[SetReferenceFields ]
Property=Reference
PropertyFieldCSVs=*/DREREFERENCE,*/URL

[IndexFields]
Property=Index
PropertyFieldCSVs=*/DREREFERENCE,*/TITLE,*/SUMMARY,*/AUTHOR

[IndexAndWeightHigher]
Property=IndexWeight
PropertyFieldCSVs=*/DRETITLE
```

```
[Properties]
0=Reference
1= Index
2= IndexWeight

[Reference]
ReferenceType=true
TrimSpaces=true

[Index]
Index=TRUE

[IndexWeight]
Index=TRUE
Weight=10
```

Note: when identifying fields you should use the format **/FieldName** to match root-level fields, ***/FieldName** to match all fields except root-level or **/Path/FieldName** to match fields that the specified path points to. If you just specify the **FieldName**, the DRE automatically adds a ***/** to it.

Using Security modules

Note: if you are running your DRE on a UNIX platform, you need to specify the LD_LIBRARY_PATH to ensure that the DRE can find the shared objects that it requires in order to implement security.

DRE configuration

If you are integrating the DRE with an application that relies on security being set up for each DRE database (for example, Portal-in-a-Box 3), you need to configure security settings as you would have done for DRE 3.

For example:

```
[Databases}
NumDBs=1
0=Archive

[Archive]
Security=NTMapped
```

In this example, the DRE contains one database called Archive. All documents that are stored in this database use the **NT Mapped** security module (please refer to **Appendix C** for details on the available security modules).

Note: you cannot have more security modules than databases.

Querying

When a user queries the DRE the appropriate security details are added to his query. If a user, for example, uses Portal-in-a-Box 3 to send the query "DNA fingerprinting, molecular evolution and molecular phylogeny", the query is sent to the DRE in the following form:

http://<IPaddress>:<query_port>/qmethod=q&querytext=DNA fingerprinting, molecular evolution and molecular phylogeny&userArchive=<username>&groupArchive=<groups_that_user_belongs_to>&domainArchive=<domain_of_user>

Using templates

The configuration file of DRE 4 / AXE contains a **[Templates]** section, which allows you to specify the templates that are required in order to output results in a manner that is compatible with non-ACI compatible applications (for example, Portal-in-a-Box 3).

The default templates that you can use are normally stored in a templates subdirectory within the DRE installation directory. However, this may vary according to the application with which you are installing the DRE.

If you want to display DRE 4/ AXE output results in a manner that is compatible with non-ACI compatible applications, you need to use the following templates in the configuration file's [Templates] section:

```
resultsheader.txt
resultsbody.txt
resultsfooter.txt
contentheader.txt
contentbody.txt
contentfooter.txt
```

For example:

```
[TEMPLATES]
0=results_template
1=content_template
DefaultResultsTemplate=results_template
DefaultContentTemplate=content_template

[results_template]
TemplateHeader=templates/resultsheader.txt
TemplateBody=templates/resultsbody.txt
TemplateFooter=templates/resultsfooter.txt
TemplateMimeType=text/plain
```

```
// Content Templates are used when the DRE responds to qmethod=C and qmethod=G
resquests

// all other query methods use the Results Templates

[content_template]

TemplateHeader=templates/contentheader.txt

TemplateBody=templates/contentbody.txt

TemplateFooter=templates/contentfooter.txt
```

Note:

- All templates have a header, body and footer template file.
- The header and the footer template is only used once, the body template is used for each result.
- You can use fields in the header template, for example to display the most relevant result.
- Templates are always in plain text, however you can configure the mime type the DRE returns for each template in the DRE configuration file.
- The DRE requires you to configure the results and the content default templates.
- If you want to select a template at query time, you need to add **&template=<template_name>** (for example **&template=content.tmpl**) to the query string.

Template tags

All template tags have the following format (like an HTML comment): **<!-- ATNMY_operator -->**

Header tags

<!-- ATNMY_HITS -->	Displays the number of items in the results list.

Body tags

<!-- ATNMY_WEIGHT -->	Displays the percentage relevance of the document.
<!-- ATNMY_NODEID -->	Displays the node ID.
<!-- ATNMY_LINKS -->	Displays a list of terms that link results to the query you entered.
<!-- ATNMY_DBASE -->	Displays the number of the database that contains the result document.
<!-- ATNMY_DBNAME -->	Displays the name of the database that contains the result document.
<!-- ATNMY_QUICKSUMMARY -->	Displays a quick summary.
<!-- ATNMY_CONCEPTSUMMARY -->	Displays a concept summary.
<!-- ATNMY_DATE -->	Displays the document's date in seconds since 1970.
<!-- ATNMY_EXPIREDATE -->	Displays the scheduled expiry date of the document as seconds since 1970.
<!-- ATNMY_NUMBER -->	Displays the position of the result document in the list of results (not normally used).
<!-- ATNMY_SECTION -->	Displays a document section number.

Tag for inserting a field

<!-- ATNMY_FIELD PREFIX="" MATCH="*/DRETITLE" SUFFIX=" " TRIM="1" ESCAPE="1" ALLSECTIONS="0" MAXFIELDMATCHES="1" -->Tags

ATNMY_FIELD	Prints every field matched by the MATCH expression within the document.
PREFIX	Allows you to enter text that you want to print before each instance of the ATNMY_FIELD.
MATCH	Enter or more strings to specify which fields ATNMY_FIELD should print. You can use wildcards in the string. By default this is ***/DRETITLE**, which means that ATNMY_FIELD prints every field that ends in **/DRETITLE**. **Note:** if you want to specify multiple fields, you need to separate them with commas (there must be no space before or after a comma).
SUFFIX	Allows you to enter text that you want to print after each instance of the ATNMY_FIELD.
TRIM	Enter **1** if you want to remove line break characters (this is in keeping with earlier DRE versions). This is the default. Enter **0** if you don't want to remove line break characters.
ESCAPE	Enter **1** if you want to escape any special characters that the ATNMY_FIELD may contain according to XML escaping rules (for example, to change **&** to **&amp;**). This is the default. Enter **0** if you don't want to escape special characters.
ALLSECTIONS	Enter **1** if you want to only print matching fields that are within the section of the result document that is displayed. Enter **0** if you want to print all matching fields the result document contains. This is the default.
MAXFIELDMATCHES	Allows you to specify the number of matching fields after which the DRE will stop printing (normally the DRE prints every matching field). This is useful if a document has multiple author fields but you only want to display the first one in the results.

Footer tags

<!-- ATNMY_ORIGINAL -->	Displays the original query string.

Using a Thesaurus DRE

In DRE 3 you were required to set up a Thesaurus DRE in order to be able to send a synonym query to the DRE. A synonym query returns results which are conceptually similar to the query's terms and / or conceptually similar to the synonyms that are available for the query's terms.

In DRE 4 / AXE you can enable synonym queries by creating a synonym file and configuring the DRE to use this file (see the **Enabling synonym queries** section in the **Configuring the DRE** chapter for details). You can then send a synonym query to the DRE, for example:

```
http://12.3.4.56:4000/action=Query&Text=Eschew obfuscation and espouse
elucidation&Synonym=true
```

Indexing configuration parameters

For details on DRE 3 indexing configuration parameters, please refer to the DRE 4 configuration help HTML file, which is located in your DRE 4 directory. To generate help, issue the following command from the command line:

<AXE_installation_directory_path>axe.exe -help.

The configuration help details all available parameters, lists deprecated parameters and provides you with information on how you can replace deprecated parameters with new settings.

Appendix C: DRE Administration dialog

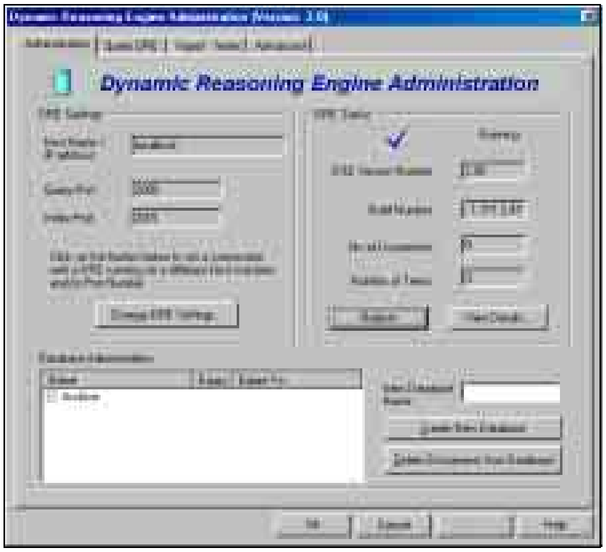
The GUI Admin is a user-friendly way of administering a DRE. It allows you to alter which DRE you are looking at, index documents into the DRE, remove documents from the DRE, query the Databases in the DRE, add new databases and delete documents from databases.

Starting up the GUI Admin

To start the GUI Admin, you can execute the program through the start menu or double click on the <InstallationName>_ContentDREadmin.exe file located in the **dre** subdirectory. Once you have started up the GUI Admin, it will automatically connect to the DRE with which it was installed and the main window will be displayed.

Note:

- You must name the DRE Administration dialog executable **<DRE_exe_name>DREadmin.exe**.
- The Import - Index page is only available if an importwizard.exe can be located in the current directory.



Using the GUI Admin

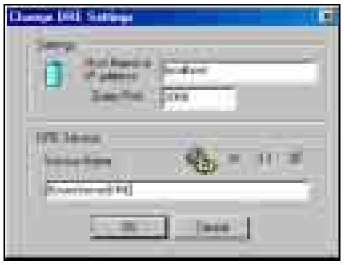
GUI ITEM	DESCRIPTION
DRE Settings	Displays the DRE Details.
DRE Status	Displays the statistics of the DRE including version and contents of the DRE.
Database List	Displays: • the Databases contained in the DRE • number of hours that documents will remain in the database after indexing • name of the database that expired documents will be moved into
“Change DRE Settings” Button	Sets which DRE you are administering
“View Details” Button	Allows you to view the last few commands sent to the DRE, both on the Query Port and Index Port.
“View DRE.CFG” Button (Under “View Details” Button	Allows you to view the DRE.CFG file of the DRE.
“Create New Database” Button	Allows you to create a new database in the DRE.
“Delete documents from Database” Button	Empties a selected Database of all its contents.

If you double-click on the databases you can see the statistics for each of the databases:

You cannot remove databases using the GUI Admin. To do this you must manually remove the database from the DRE configuration file. Also, it is not possible to set databases to expire from here. You will need to edit the DRE configuration file.

Changing DRE settings

If you want to change the DRE settings click on the “Change DRE Settings” button and edit the settings as necessary. From here you can also specify the name of the service corresponding with the DRE executable and start and stop the service using the equivalent buttons.

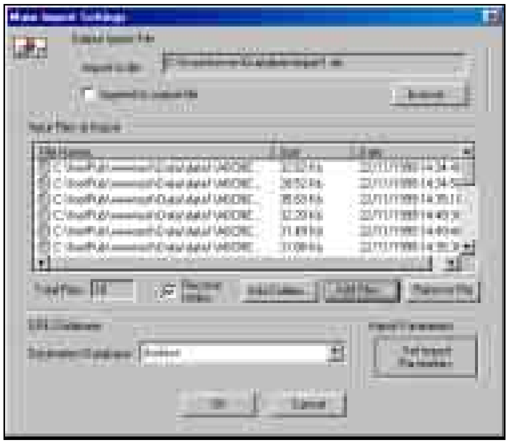


To change these settings the DRE must be local. The changes can be made if an NT Server is used and the full path is given to the DRE configuration file as an argument.

Importing files into an .IDX File

It is only possible to index .IDX files into the DRE. If you want to index any other type of file then you have to import it first. If you are importing files other than in HTML or PDF format you must have the Omnislave .DLL files together with the Omnislave.exe executable file installed in your working directory. For .PDF files you must have the pdfslave.exe executable file installed in your working directory. HTML and Text files require no extra files for importing.

To import files click on the “Import” button in the main GUI Admin window. Add the files to the “Input files to Import” list by clicking on the “Add” button. Alternatively you can drag and drop files onto the “Input Files to Import” list. Select the database in which the files are to be indexed into and once all of the criteria have been entered, click on the “Next” button.



GUI ITEM	DESCRIPTION
Input Files to Import	Lists the files to import into an IDX file
“Add Files” Button	Use to add files to the list.
“Add Folders” Button	Use to add directories to the list. Tick on the “Recurse Folder” box if you want to recurse subdirectories.
“Remove File” Button	Use to remove files/directories from the list.
Output Import File	.IDX filename into which list of files will be imported.
DRE Database	The database into which the IDX file will be indexed in the DRE.
“Set Import Parameters” Button	Launches the Import wizard enabling you to specify various criteria according to which the documents should be imported into the idx format.

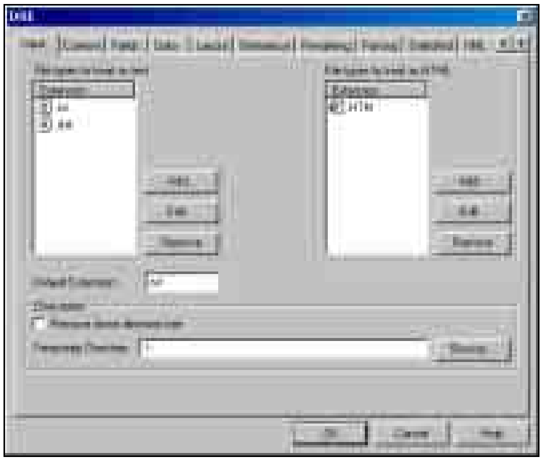
Using the Import Wizard

The Import Wizard is used to specify the idx format corresponding to each document. It consists of multiple tabs used for different settings which control how the file will be generated during importing. These sets of options are:

Input	On the left of the input page, you can select the extensions of files to be treated as text and on the right you can select the extensions of files to be treated as HTML.
Content	The content can be stored in three different ways. All of these are possible at once. These are: • the plain text content • a summary • intelligent title and summary The number of sentences for a summary can be specified. This property sheet also allows you to specify the criteria that the document must meet for it to be imported. You can specify the minimum and maximum size of the document to be imported in both words and bytes. If you would like large documents to be broken up into a number of paragraphs, then you must check the breaking checkbox on this page. You can then specify the size that a document must be for it to be broken up and the size of the paragraphs produced.

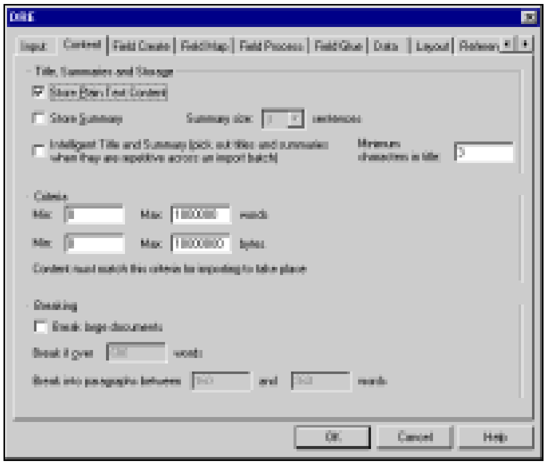
Field Create	Specifies the fields that are to be stored in the DRE. These are both fixed fields and dynamic fields. For further details, please see the various field sections below.
Field Map	Specifies the mapping of fields within the DRE. See below
Field Process	This enables processes to be specified for a field set up in the DRE with parameters attached to that process. For example extract part of the field starting at a certain character.
Field Glue	This specifies fields that can be glued together or a field that can have a string glued to it.
Data	Specifies certain file data (date and length) that are to be extracted from the documents and stored in fields in the DRE. The details of the way in which data is to be extracted is specified here. These details include where to extract data from, where to extract it to and in what format. These details are explained further in the data section below.
Layout	Specifies what content in the documents is to be stored. You must define the beginning and end of documents. This page includes HTML start and end definitions and text start and end definitions. The HTML start and end definitions are used before the content has been converted into plain text and the text start and end definitions are used after wards. You must also specify the position of page breaks on this sheet.
Reference	Specifies reference replacements for documents imported. This includes the part of the path to replace and what to replace it with. It is also on this property sheet that you must specify how to truncate a path.
Rendering	Specifies how to treat the temporary HTML documents generated when processing omnislave/pdf imported documents.
Parsing	Specifies data that should be stripped from the imported document.
Delimited	Specifies how to deal with delimited files. You must specify the files to be treated as delimited by listing the extensions of those files. You must also specify document start and end delimiters and field start and end delimiters.
XML	Specifies how to deal with XML files. You must list the extensions of files to be treated as XML in the box provided.
Custom Slaves	Specifies extra custom made slaves that are needed to import certain file types. You must give your newly created slave a name and enter the file type that it is to import. The type will then be associated with that slave. You can have more than one associated file type for each slave but you must make a separate entry for each file type i.e. you will have to enter your new slave more than once too.
Advanced	Specifies miscellaneous details that can be applied in the importing process. These details are explained in the Advanced section below.

Input



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

Content



Title, Summaries and Storage

[✓] Store Plain Text Content

[ ] Store Summary — Summary size: [illegible] sentences

[ ] Intelligent Title and Summary (pick out titles and summaries when they are repetitive across an import batch) — Minimum characters in title: 3

Criteria

Min: 0 Max: [illegible] words

Min: 0 Max: [illegible] bytes

Content must match this criteria for importing to take place

Breaking

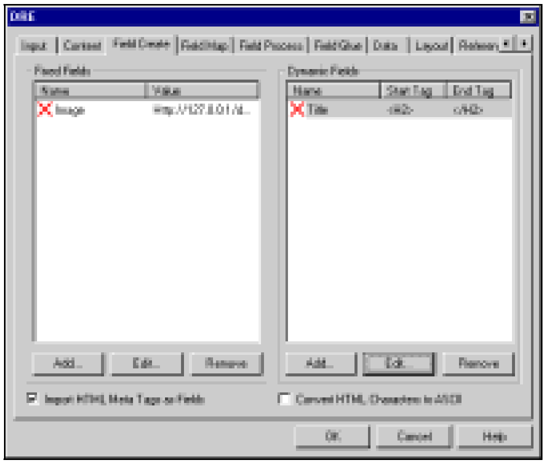
[ ] Break large documents

Break if over [illegible] words

Break into paragraphs between [illegible] and [illegible] words

Parameter	Description
Store Plain Text Content	Specifies whether or not the text content of the documents are to be imported
Store Summary	Specifies whether or not the first n lines of the documents are to be imported as a summary.
Intelligent Title and Summary	Specifies whether or not a unique intelligent title and summary are to be generated for the documents that are imported.
Minimum Characters in Title	This specifies the minimum amount of characters that can be used as a title. If a title of less than 3 characters is found the title will be taken from the content of the document.
Criteria – Min/Max Words	Indicates that documents with less than Min words and more than Max words are not to be imported.
Criteria – Min/Max Bytes	Indicates that documents with less than Min Bytes and more than Max Bytes are not to be imported.
Breaking – Break large documents	Specifies whether or not documents over a certain size should be broken up into smaller sections for easier indexing.
Breaking – Break if over	If documents are over this number of words, then the document is broken up into smaller sections for easier indexing.
Breaking – Break into paragraphs between	When documents are broken up into smaller sections, the sections have at least n paragraphs and at most m paragraphs.

Field Create



Fixed Fields

Name	Value
Image	http://127.0.0.1/d...

Dynamic Fields

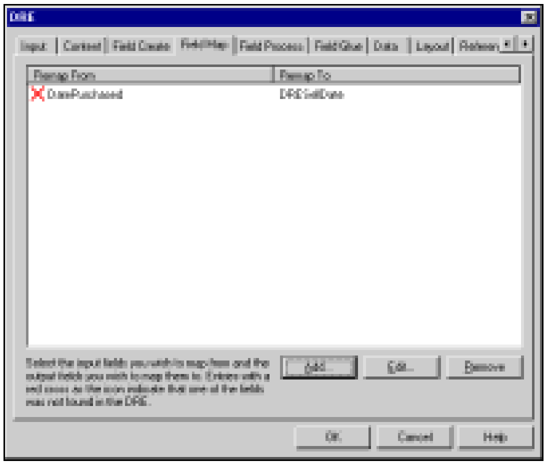
Name	Start Tag	End Tag
Title	<H2>	</H2>

☑ Import HTML Meta Tags as Fields

☐ Convert HTML Characters to ASCII

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

Field Map

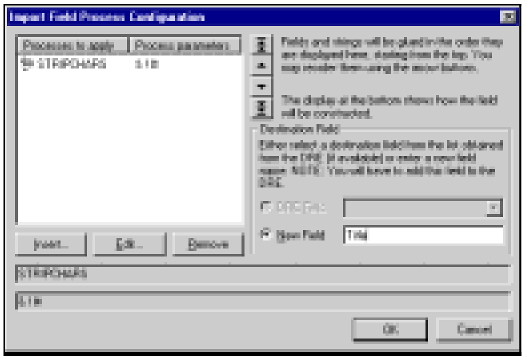


Remap From	Remap To
DatePurchased	DRESellDate

Select the input fields you wish to map from and the output fields you wish to map them to. Entries with a red cross as the icon indicate that one of the fields was not found in the DRE.

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

Field Process



STRIPCHARS	0,1,0

Fields and strings will be glued in the order they are displayed here, starting from the top. You may reorder them using the arrow buttons.

The display at the bottom shows how the field will be constructed.

Either select a destination field from the list obtained from the DRE (if available) or enter a new field name. NOTE: You will have to add this field to the DRE.

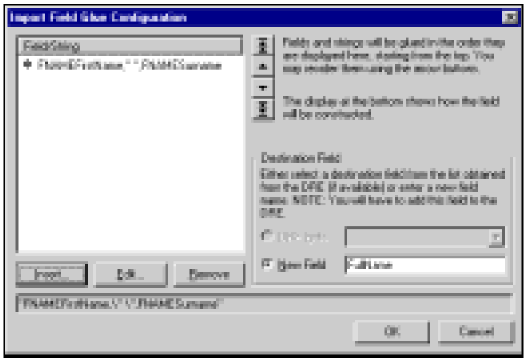
New Field: Title

STRIPCHARS

0,1,0

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

Field Glue


Import Field Glue Configuration

Field/String

FNAME<FirstName>," ",FNAME<Surname>

Fields and strings will be glued in the order they are displayed here, starting from the top. You may reorder them using the arrow buttons.

The display at the bottom shows how the field will be constructed.

Destination Field

Either select a destination field from the list obtained from the DRE (if available) or enter a new field name. NOTE: You will have to add this field to the DRE.

DRE fields:

New Field: FullName

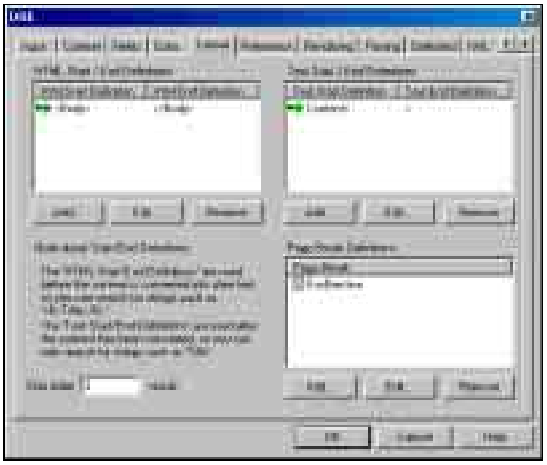
FNAME<FirstName>," ",FNAME<Surname>"

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

Data

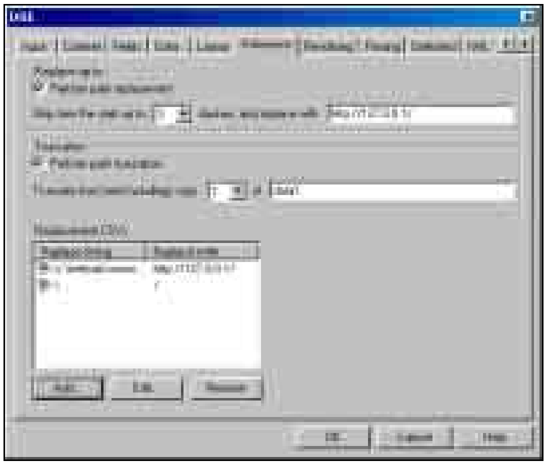
Parameter	Description
Date – Extract From	Indicates from where in the File data a date is to be extracted. This can be either: **Now** Current date **Accessed** Date it was last accessed **Created** Date it was created **Modified** Date it was last modified **Field** Date taken from a specific file field **Content** First date found in the content of the document **Filename** First date found in the file name
Date – Format	Specifies the formats in which the required date can be found.
Date – Extract from field	Specifies the name of a date field in the DRE whose value is to be extracted.
Date – Extract to field	Specifies the name of the DRE field where the date extracted is to be stored.
Date – Extract to format	Specifies the format that the extracted date will be stored as in the DRE. If you are extracting to the field DREDATE, then the format must be yyyy/mm/dd or number of seconds since 1970 or NOW
Length – Extract from	Specifies from where the length of the document is to be extracted. Currently this can only be “File”.
Length – Extract to field	Specifies the name of the field in the DRE where the length of the document is to be stored.

Layout



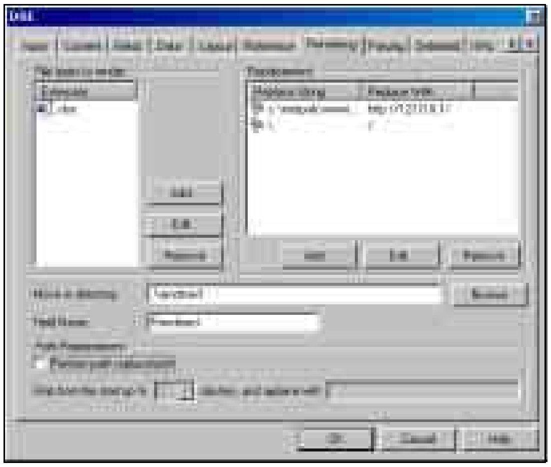
Parameter	Description
HTML Start/End Definitions – HTML Start Definition	A string of characters that mark the start of the content to be imported in HTML documents.
HTML Start/End Definitions – HTML End Definition	A string of characters that mark the end of the content to be imported in HTML documents.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

Reference



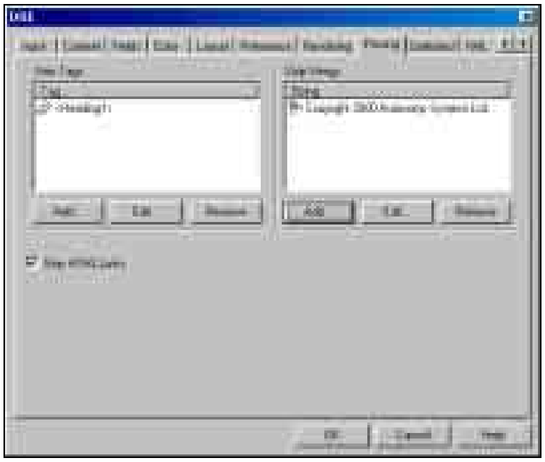
Parameter	Description
Replace up to – Perform Path Replacement	Specifies whether or not the string up to a certain \ in the file reference is to be replaced.
Replace up to – Strip from the start up to…	Specifies the nth slash where the preceding string is replaced with the string specified.
Replace up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

Rendering



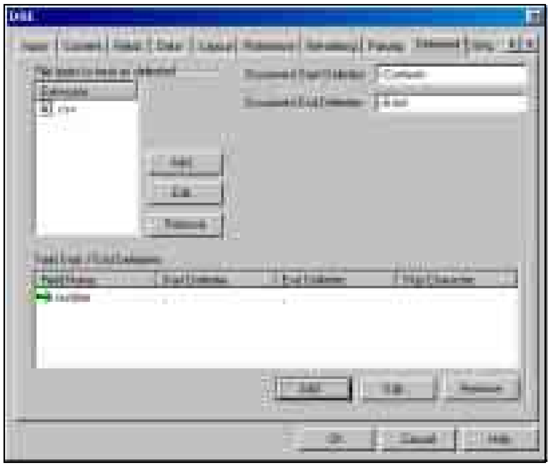
Parameter	Description
File types to render – Extension	Specifies the extension of files on which to perform HTML rendering
Replacement – Replace string	Specifies the string that is to be replaced. Used to set the right reference for the Field Name field.
Replacement – Replace with	Specifies the string that is to replace the original reference of the ImportRenderedHtmlFieldName field. Used to set the right reference for the Field Name field.
Move to directory	For any non-HTML file that gets imported, the HTML file that comes out of omnislave/PDF gets moved into the specified directory.
Field Name	For any non HTML file that gets imported, the HTML file that comes out of omnislave/PDF gets moved into the this DRE field.
Path Replacement – Strip from the start up to …	Specifies a string that is to be replaced up to a certain ‘\’. Used to set the right reference for the Field Name field.
Path Replacement – …replace with…	Specifies the replacement string of the string immediately preceding the nth ‘\’.

Parsing



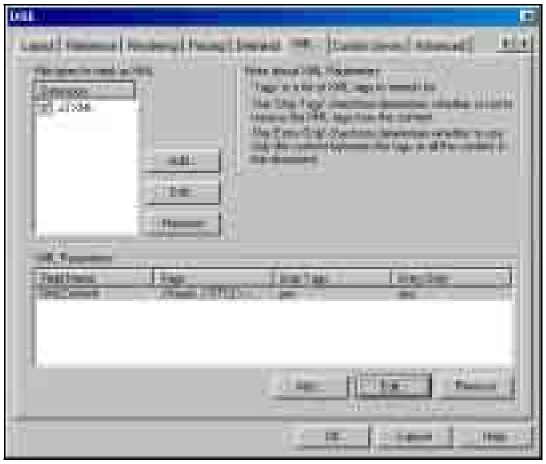
Parameter	Description
Strip Tags	Strips the content between begin and end tags. E.g. ImportStripTagCSVs=<B>, will strip all content between <B> and <\B>.
Strip Strings	Comma separated list of strings that, when matched in a document, are to be stripped from the idx format.
Strip HTML Links	Specifies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

Delimited



Parameter	Description
File types to treat as delimited	Specifies the extension of the files to treat as delimited files.
Document Start Delimiter	In the case of importing a delimited file, e.g. a .CSV file, this specifies the beginning of a file.
Document End Delimiter	In the case of importing a delimited file, e.g. a .CSV file, this specifies the end of a file.
Field Start/End Delimiters – Field Name	Specifies the name of the field where data from a delimited file is stored.
Field Start/End Delimiters – Start Delimiter	Specifies the start string of data to be inserted into the idx field denoted by Field Name.
Field Start/End Delimiters – End Delimiter	Specifies the end string of data to be inserted into the idx field denoted by Field Name.
Field Start/End Delimiters – Skip Character	Specifies the number of characters to skip from the beginning of a document when producing the idx file.

XML

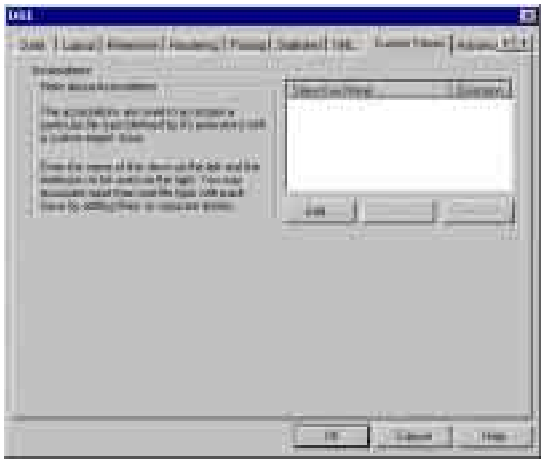


Parameter	Description
File types to treat as XML	A list of all XML file extensions.
XML Parameters – Field Name	Will specify the field in the DRE where the content taken from the XML File is stored.
XML Parameters – Tags	Searches for the appropriate XML tag(s). This can be a hierarchy level of search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

Parameter	Description

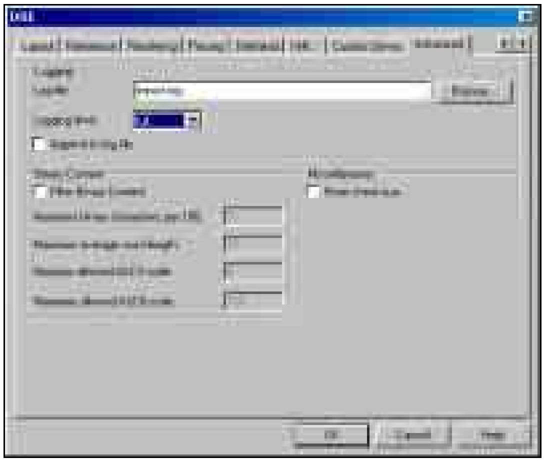
XML Parameters – Entry Only	Only imports the content between the searched tags at that level.
	
	For example if an XML file contained the following strings:
	
	<TAG1>
	  <TAG3>
	    Some Other Tag3 entry
	  </TAG3>
	  <TAG2>
	    Some data
	    <TAG3>
	      The entry for Tag3
	    </TAG3>
	  </TAG2>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

Custom Slaves



Parameter	Description
Slave Exe Name	Name of the custom slave executable.
Extension	Extension of files that are to be processed by the custom slave executable.

Advanced



Parameter	Description
Log File	Name to be given to the import log file
Logging Level	This is the type of logging that is to be performed. This can be either: • Warnings • Errors • Full
Append to log	Specifies whether or not import logging for this session should be appended to an existing log file.
Binary Content – Filter Binary Content	Specifies whether or not binary content is to be processed.
Binary Content – Maximum binary characters per 100	Maximum number of binary characters tolerated per 100 characters. Anything binary exceeding this is to be ignored.
Binary Content – Maximum Average Word length	Binary words with more than this number of characters will not be imported.

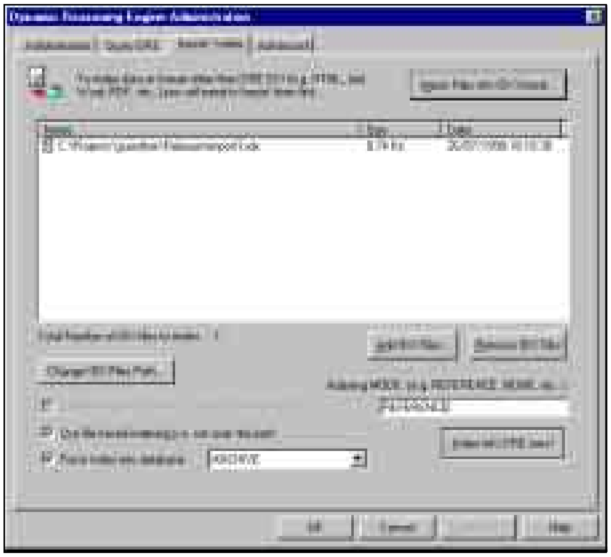
Parameter	**Description**
Binary Content – Minimum allowed ASCII code	The minimum ASCII value allowed in characters in the binary content. Anything lower is treated as binary and removed.
Binary Content – Maximum allowed ASCII code	The maximum ASCII value allowed in characters in the binary content. Anything higher is treated as binary and removed.
Store Checksum	When checked, a value is added to the Checksum field in the [Field] section in the DRE.CFG. This field value is used to determine whether or not to show a document result in the front-end. This field is typically used with Autonomy Update.

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

Indexing IDX Files Into The DRE

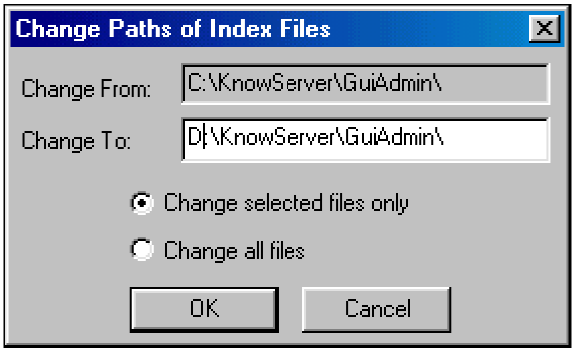
When indexing .IDX files into the DRE, you can only force the .IDX file to be indexed into the specific database when the “Use file based indexing” checkbox has been marked. If you do not force indexing into a specific database, the file will be indexed into the database specified when the IDX file was created. If your specified IDX file is not already included in the “Files To Index” list, add the IDX files to the list by clicking on the “Add” button and browsing for the required file. Alternatively you can drag and drop the .idx files onto the list. At this point if you wish to view and/or edit a particular .IDX file, double clicking on the listed file path will launch MS Write and display the file.

If you are indexing a large file over the socket, the GUI Admin will ask you if you are sure that this is what you want to do.



Parameter	Description
Delete Files after indexing	This can only be executed if indexing over the port. It will automatically delete all idx files after indexing has taken place.
Use file based indexing	This can only be used if the DRE GUI Admin and the indexing DRE resides on the same machine. The idx file is not sent over the port.
Force index into database	This can only be executed if using file based indexing. The documents to be indexed in the DRE are forced into the specified database as opposed to the database specified in the idx file.
Indexing MODE	Specifies how documents are to be replaced in the DRE.

Select the extra features that you wish to be performed during/after the indexing process by marking the appropriate check boxes. If you are doing a file based index, you will need to change the Index File Path so that the path is with respect to the DRE into which it is being indexed. To do this, click on the “Change Index Files Path.” button and alter the path appropriately.


Change Paths of Index Files

Change From: C:\KnowServer\GuiAdmin\

Change To: D:\KnowServer\GuiAdmin\

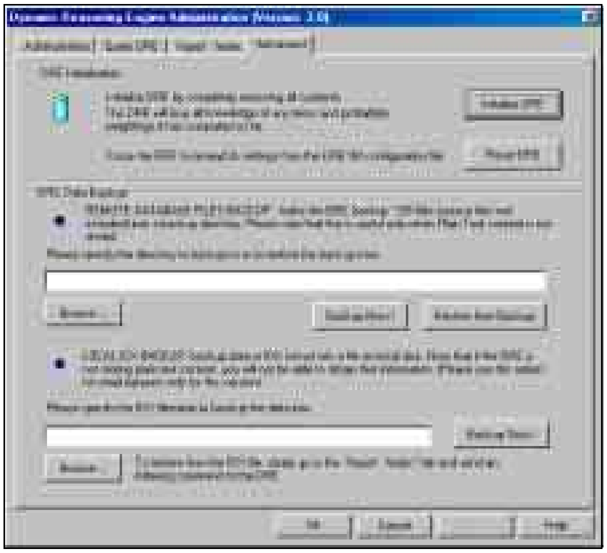
[✓ Change selected files only] Change all files

If you select “Change All Files” then the paths of all files listed will be changed otherwise just the selected path will be altered.

Once you have added all the IDX files that you wish to index and have given the required criteria, click on the “Index to DRE Now!” button. The file will then be added to the DRE.

Advanced Settings

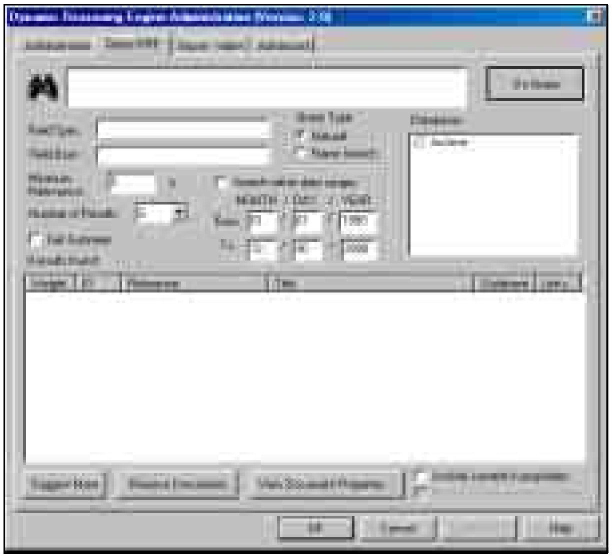
This screen contains advanced settings for administering the DRE, and backing up the data within the DRE.



GUI ITEM	DESCRIPTION
DRE Initialization – Initialize DRE	Initializes the DRE, which deletes all documents as well as all terms and weights in the Engine.
DRE Initialization – Reset DRE	Forces the DRE to re-read its settings from the DRE.CFG file.
DRE Data Backup	• Remote database backup: sends a command to the DRE for it to save all its database files into a directory (remember the path must be with respect to the DRE). You can restore from that database as well. • Local IDX backup: it sends a command to the DRE so that it saves all its data in IDX format into the file specified. To restore from this IDX file, simply index it from the Index Tab.

Querying The DRE Through The GUI Admin

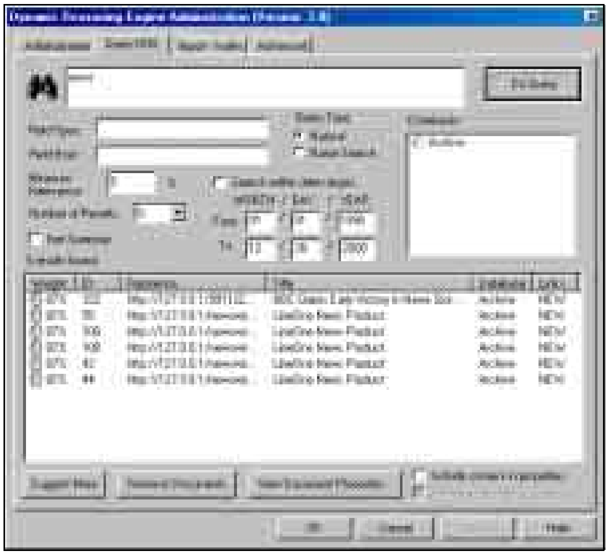
To query the DRE, click on the “Query” tab.



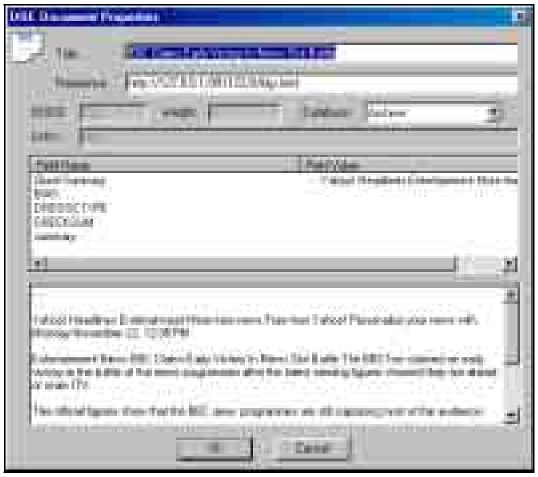
GUI ITEM	DESCRIPTION
Field Name	This specifies field restriction for the query. E.G. fnameTITLE=ANY(money)
Field Bool	This specifies how the fields are restricted with respect to each other. E.G. fnameTITLE+AND+fnamePRICE
Query Type	Natural means a natural language search. Proper Name search is where Proper Names, such as John Smith get treated as one compound word.
“Get Summary” check box	This will obtain the first few lines as a summary field which will be shown in the Document Properties window
Minimum Relevance	Obtains results that have at least this relevance value
Number of Results	Number of results to return

GUI ITEM	DESCRIPTION
Search within Date Ranges	Restrict results to a data range
Databases List	List of available databases. When doing a query you can select only the ones you want to query for. If none are selected, all of them will be queried.
“Suggest More” Button	Suggest documents related to the selected result(s).
“Remove Documents” Button	Remove selected document(s) from result list.
“View Document Properties” Button	View the properties of select document. Properties include, field values, date, doc id, relevance, etc..
“Include Content In Properties” tick box	Includes the plain text content in the Document Properties window.
“Re-index changes to Properties”	When changes are made in the Document Properties, it will re-index the changes into the DRE. NOTE: This can only be done when you are storing plain text content in the DRE.

Enter the appropriate criteria for your query and click on the “Do Query” button. Your results will then be displayed in the Results table as shown below.



From here it is possible to view the content of each result returned. If the reference starts with http:// then the GUI Admin will launch your default browser when you double click on the result. Otherwise, simply select the required document, make sure that the “Include content in properties” check box is checked and click on the “View Document Properties” button:



If you have marked the “Re-index changes to properties” checkbox in the Query DRE Window, you will be able to edit the contents of the document and this will then be saved. As well as editing results it is also possible to remove results from the DRE. Simply select the appropriate document and click on the “Remove from DRE” button.

Also, from the Query DRE Window, it is possible to suggest further documents based on a selected result. To do this select the appropriate result in the Results box and click on the “Suggest More” button. These results will then be displayed in the Results box and again it is possible to perform the above stated functions on these results.

Appendix D: Boolean operators

When you are using the following query command, you can specify the query text using a Boolean expression:

http:// <IPAddress>:<QueryPort>/action=Query&Text=<QueryText>

You can use the following operators to manipulate a query by applying them to words, exact phrases or other Boolean expressions. Note that APCM will still be used to rank the results that match the Boolean query.

AND

Binary operator. Ensures that both terms are matched in every document that is returned.

For example:

action=Query&Text=cat+AND+dog

This query only returns documents that contain both “cat” and “dog”.

NOT

Unary operator. Ensures that the term following NOT is excluded from any of the returned documents.

For example:

action=Query&Text=cat+NOT+dog

This query only returns documents that contain "cat" but not "dog".

OR

Binary operator. One or both terms must appear for the document to be returned. This is the default behavior if no explicit operator is given between two terms.

For example:

action=Query&Text=cat+OR+dog

This query only returns documents that contain either “cat”, “dog” or both terms.

EOR

or

XOR

Binary operator. Logical exclusive OR. Only one of the terms is permitted to appear for the document to be returned. This is a rarely used operator.

For example:

action=Query&Text=cat+XOR+dog

This query only returns documents that contain either the term “cat” or the term “dog”. Documents that contain both "cat" and "dog" are not returned.

NEAR*nn* Only returns documents in which the second term is within ***nn*** words of the first term.

For example:

action=Query&Text=cat+NEAR1+dog

This query only returns documents in which the term “cat” is no more than 1 word away from “dog”. This means that documents, which contain "cats and dogs" and documents that contain "dogs and cats" are returned, while documents that contain "cats do not like dogs" are not returned (as the terms are not close enough to each other).

DNEAR*nn* Directed NEAR. Only returns documents in which the second term is within ***nn*** words of the first term, in the specified order.

For example:

action=Query&Text=cat+DNEAR1+dog

This query only returns documents in which the term "dog" follows the term "cat", but no more than one word away from the term "cat". This means that documents, which contain "cats and dogs" are returned, while documents that contain "dogs and cats" or "cats do not like dogs" are not returned.

WNEAR*nn* Weighted NEAR. Proximity operator that promotes relevance when term spacing is less than the specified word distance (closer together implies higher relevance).

For example:

action=Query&Text=dog+WNEAR6+cat

In this query extra relevance is given to documents in which "cat" and "dog" appear within 6 words of each other in a piece of text. This weight increases as the terms get closer to each other.

BEFORE Only returns documents in which the first term precedes the second one.

For example:

action=Query&Text=cat+BEFORE+dog

This query only returns documents in which the term "dog" appears later than the term "cat".

AFTER Only returns documents in which the first term appears later than the second one.

For example:

action=Query&Text=cat+AFTER+dog

This query only returns documents in which the term "cat" appears later than the term "dog".

( )	Bracketed expressions. These are evaluated left to right and can be nested. They dictate the precedence and behavior of combined operator statements.

For example:

action=Query&Text=(fish EOR pie) AND (chips EOR mash)

This query only returns documents that contain one of the following:

"fish” and “chips”

“fish” and “mash”

“pie” and “chips”

“pie” and “mash”

Precedence of Boolean operators

Boolean operators have the following precedence:

Highest precedence:	NOT
Lower precedence:	AND; WNEAR; NEAR
Lowest precedence:	OR; XOR; EOR

Operators that have the same level of precedence have neither left or right associativity. You can use brackets to bind terms together as appropriate.

Soundex keyword search

If you are querying for a particular term, you can use Soundex to return documents that contain the same term or a phonetically similar term (using the Soundex algorithm). You can combine a Soundex keyword search with any other query form.

For example:

action=Query&Text=albert+SOUNDEX(einstine)

Exact phrases

You can search for exact phrases by putting quotation marks (" ") around a string of words. For example: “world market”

Relevance ranking

In evaluating all types of queries, the DRE employs complex algorithms based on a combination of Information Theory and Bayesian methods to weight and rank the document returns by statistical relevance. In doing so it makes use of information theoretic values calculated dynamically for all concepts on indexing, allowing relevance to be evaluated both as a percentage, and in the case of agents, such as those created by the Categorizer, as absolute values.

In practice, the relevance can be seen as a measure of the conceptual overlap between the query text and the text within a document. This can be affected in several ways; certain fields can be given extra weight by associating a weighting factor with them at indexing time. For example, extra weight can be given when query terms appear in a document's title as opposed to the body of the text.

Alternatively, relevance can be increased through use of the "WNEAR" operator; a query of "cat+WNEAR10+dog" gives extra weight to documents in which the terms "cat" and "dog" are mentioned near to one another within the text.

Appendix E: using ACI templates

You can change the way in which responses to action commands that you send to ACI Servers are displayed by writing your own template for an action. In order to use ACI templates, you must perform the following steps:

1. Write your template (please refer to the **Writing an ACI template** section of this chapter for details of how to do this).
2. Save the file to an **acitemplates** subdirectory of the main directory for your ACI Server with the extension **.tmpl**.

   Alternatively, you can upload the template to your ACI Server by submitting a Multipart request to the server with the **TemplateUpload** action. (The recommended way to submit requests that use Multipart encoding is via the ACI API.)

You can now send the action command for which you have created this template to the ACI Server. Use the action's **Template** parameter to specify the name of the template you have created in order to apply this template to the action's response.

For example, if your ACI Server is a DRE and you have written a template for the **Query** action, you can apply the template to this action using the following command:

```
http://<IP_address>:<port>/action=Query&Text=my_text&Template=MyQueryTemplate
```

The response for the **Query** action is displayed in the way you specified in the **MyQueryTemplate** template.

Note: if you alter the template after using it for an action, you need to refresh the template the ACI Server uses by modifying the command in the following way:

```
http://<IP_address>:<port>/action=Query&Text=my_text&Template=MyQueryTemplate&
forcetemplaterefresh=true
```

Writing an ACI template

Each template can comprise text and template tags. Template tags determine which details are displayed (the details that can be displayed depend on the action for which you are writing the template). The position of a template tag determines where this tag's detail is displayed.

You can edit template tags by modifying which fields (the details that the tags display in a template are stored in ACI Server fields) the tags refer to and by using operators in the tags.

Template tags can have one of the following formats:

<autn_tag:*[Operator_string]field_name*>

<autn_foreach:*field_name*>

autn_tag

Allows the ACI Server to identify the template tag and display the **field_name** field that it refers to. If you have specified an ***[Operator_string]***, the ACI Server manipulates the field as specified by this string before displaying it.

autn_foreach

Allows the ACI Server to loop through each **field_name** field that the **autn_foreach** tag refers to, in order to identify the template tag and display the fields that subsequently listed tags refer to.

Note: you must mark the end of the template tags list that you want to apply to the **autn_foreach** tag, with a closing tag. The closing tag has the format **</autn_foreach:autn:*field_name*>**.

[Operator_string]

You can use the following operators to manipulate the field that the **autn_tag** tag refers to (this is optional):

[IF <condition>]

Allows you to set a condition under which an instruction should be followed. You must mark the end of the conditional section with **[ENDIF]**.

You can use the following conditions:

-eq	equals
-ne	does not equal
-lt	less than
-le	less than or equal to
-gt	greater than
-ge	greater than or equal to

For example:

```
<autn_tag:[IF -eq 0]autn:numhits>
        Results:
        You have no results
<autn_tag:[ENDIF]>
```

In this example the text "**Results: You have no results**" is displayed if the **numhits** field contains the value **0** (that is if no results have been returned).

```
<autn_tag:[IF -gt 0]autn:numhits>
        Result
        <autn_foreach:autn:hit>
            <autn_tag:autn:title>
            <autn_tag:autn:id>
        </autn_foreach:autn:hit>
<autn_tag:[ENDIF]>
```

In this example, if the **numhits** field contains a value that is greater than **0** (that is one or more results have been returned), the text "**Result**" is displayed along with the title and ID for each result.

[ENDIF]

Marks the end of the condition labeled with the **[IF <condition>]** operator.

[COUNT]

If you are using the **<autn_foreach: …>** tag, for example to loop through all results or all agents, the **[COUNT]** operator allows you to display the number of times that the loop has been performed.

For example:

```
<autn_foreach:autn:hit>
Result <autn_tag:[COUNT]autn:hit>
```

In this example each result (**hit**) that is returned is counted (that is the first result is displayed with the number 1, the second with the number 2 and so on.

[DATEFORMAT <dateformat>]

If a tag refers to a field that contains a date, **[DATEFORMAT]** allows you to convert this date to the specified **<dateformat>**.

For example:

Date <autn_tag:[DATEFORMAT DD-SHORTMONTH-YY HH:NN:SS]autn:startdate>

In this example the date in the **startdate** field is converted to the format **DD-SHORTMONTH-YY HH:NN:SS** (for example, 04-Jan-02 08:46:37).

[ESCAPE]

Escapes the text content of the field that the tag refers to.

For example:

<autn_tag:[ESCAPE]autn:title>

In this example the document reference in the **title** field is escaped.

[UNESCAPE]

Unescapes the HTML content of the field that the tag refers to.

For example:

My Unescaped Field <autn_tag:[UNESCAPE]autn:MyField>

In this example the HTML content in the **MyField** field is unescaped.

[ENCRYPT]

Encrypts the content of the field that the tag refers to.

For example:

<autn_tag:[ENCRYPT]autn:pwd>

In this example the password in the **pwd** field is encrypted.

field_name

Enter the name of the field that you want to display. You can use **[Operator_string]** operators to manipulate the field before the ACI Server displays them.

ACI templates are action-specific. If you are uncertain which fields are available for the action for which you want to use your template, run an action command from your web browser to your ACI Server for the action for which you are creating the template.

For example, if your ACI Server is a Classification Server, you can send the following query:

http://<IP_address>:<port>/action=CategoryQuery&Category=<category_id>

The XML that this query returns contains all the fields in the result documents that are returned for the **CategoryQuery** action:

```
<autnresponse>
  <action>CATEGORYQUERY</action>
  <response>SUCCESS</response>
  <responsedata xmlns:autn="http://schemas.autonomy.com/aci/">
    <autn:numhits>2</autn:numhits>
    <autn:hit>
      <autn:title>More German towns dig in as flood wave rolls on</autn:title>
      <autn:url>http://c.moreover.com/click/here.pl?z45255344&z=161380</autn:url>
      <autn:id>49273</autn:id>
      <autn:weight>81</autn:weight>
      <autn:links>FLOOD,GERMANI,GLOBAL,WARM,BLAME</autn:links>
      <autn:database>0</autn:database>
      <autn:summary>By Arnd Wiegmann BITTERFELD, Germany, Aug 18 (Reuters) -
      German emergency workers toiled on Sunday to patch leaking dykes to try to save
      homes, crops and chemical factories from a flood wave that has ravaged swathes
      of central Europe, killing almost 100 people. At a summit in Berlin, the European
      Commission agreed with leaders from Germany, Austria, the and on an aid
      package to help repair the devastation wreaked by record floods over the last
      week</autn:summary>
      <autn:contextsummary>Floods and landslides caused by torrential rains have also
      killed at least 133 people in China in recent days. Many environmentalists blame
      the extreme weather on global warming and have urged a tougher fight against
      pollution.</autn:contextsummary>
    </autn:hit>
    <autn:hit>
      <autn:title>Soggy Europe battles raging rivers</autn:title>
      <autn:url>http://c.moreover.com/click/here.pl?z45012660&z=161380</autn:url>
      <autn:id>17968</autn:id>
      <autn:weight>78</autn:weight>
      <autn:links>FLOOD,GERMANI,GLOBAL,WARM,BLAME</autn:links>
```

```
            <autn:database>0</autn:database>
            <autn:summary>By Karel Janicek / The Associated Press Prague - Firefighters and
            volunteers fought back river waters swamping the National Theatre and other
            landmarks Wednesday as unprecedented flooding that has killed at least 98 people
            across Europe threatened this medieval city's historic Old Town. The raging Vltava
            River hit its highest point yet in the flooding, just hours after police evacuated
            residents from the capital's quaint cobblestone centre</autn:summary>
            <autn:contextsummary>Flood waters gradually began receding late Wednesday,
            raising spirits among officials who worked around the clock to secure the city.
            Germany, also the scene of devastating flooding, reported 10 deaths, with six other
            people missing.</autn:contextsummary>
        </autn:hit>
    </responsedata>
</autnresponse>
```

In this example you can see from the XML that Classification Server returns that the following fields are available:

autn:numhits

autn:hit

autn:title

autn:url

autn:id

autn:weight

autn:links

autn:database

autn:summary

autn:contextsummary

Example template

The following template has been created for Classification Server's **CategoryQuery** action.

```
 <autn_tag:[IF -gt 0]autn:numhits>
     category results:
     <autn_foreach:autn:hit>
         result number: <autn_tag:[COUNT]autn:hit>
         result document title: <autn_tag:autn:title>
         result document weight: <autn_tag:autn:weight>%
         <autn_tag:autn:summary>
     </autn:foreach>
<autn_tag:[ENDIF]>
<autn_tag:[IF -eq 0]autn:numhits>
     There are no results for this category
<autn_tag:[ENDIF]>
```

You can use this template with the **CategoryQuery** action, by entering the following command:

```
http://<IP_address>:<port>/action=CategoryQuery&Category=<category_number>&Tem
plate=<template_name>
```

In response Classification Server returns the following:

```
<autnresponse>
    <action>CATEGORYQUERY</action>
    <response>SUCCESS</response>
    <responsedata xmlns:autn="http://schemas.autonomy.com/aci/">
        <autn:templatedata>category results: result number: 1 result document title: More
        German towns dig in as flood wave rolls on result document weight: 81% content: By
        Arnd Wiegmann BITTERFELD, Germany, Aug 18 (Reuters) - German emergency
        workers toiled on Sunday to patch leaking dykes to try to save homes, crops and
        chemical factories from a flood wave that has ravaged swathes of central Europe,
        killing almost 100 people. At a summit in Berlin, the European Commission agreed with
        leaders from Germany, Austria, the and on an aid package to help repair the
        devastation wreaked by record floods over the last week ---------- result number: 2
        result document title: Soggy Europe battles raging rivers result document weight: 78%
        content: By Karel Janicek / The Associated Press Prague - Firefighters and volunteers
        fought back river waters swamping the National Theatre and other landmarks
        Wednesday as unprecedented flooding that has killed at least 98 people across Europe
        threatened this medieval city's historic Old Town. The raging Vltava River hit its highest
        point yet in the flooding, just hours after police evacuated residents from the capital's
        quaint cobblestone centre ---------- </autn:templatedata>
    </responsedata>
</autnresponse>
```

Appendix F: Error codes

Internal errors

3113	4113	9114	18003
3114	4114	9115	18005
3115	4115	13002	18006
4103	4117	13003	
4104	9113	18002	

Error codes

201	Invalid language type
	The language type entered in the query is invalid
1001	No valid terms provided
2001	No valid document ids or references supplied
3001	No query text supplied
3002	SUGGESTONTEXT: The query text contained no terms that appear in the DRE
4001	No query text supplied
4002	No valid query terms
4101	Out of memory
4102	Invalid Field Text
4105	Out of memory
4106	Bracket Mismatch in the query text
4107	Incorrectly formed Boolean expression in the query text
4108	Incorrectly formed Boolean expression in the query text
4109	Out of memory
4110	Bracket Mismatch in the field text
4111	Incorrectly formed Boolean expression in the field text

4112	Incorrectly formed Boolean expression in the field text
4116	Incorrectly formed Boolean expression in the field text
4118	No valid query terms
4121	Field Tree parsing error
	QUERY: There was an error building the Boolean field tree. This could potentially be caused by having a single field specifier that has too many characters, if there are any with many hundreds of characters, try reducing them in length.
8001	No valid terms supplied
9001	No valid documents or text supplied
9116	Invalid language type or encoding
	QUERY: The language type or > output encoding given in the query is invalid
10001	No valid documents ids or references supplied
11001	System out of memory
11002	No terms found
13001	No valid documents or text supplied
13004	Type of summary not specified; summary=concept/context/quick
15001	The value of MaxResults is invalid
15002	No valid documents were found
16001	Highlight - Invalid parameter
16002	Internal error in highlighting
17001	No documents found
18001	No valid query text
18007	Unlicensed request
	This DRE is not licensed to perform automatic language detection, is not initialized correctly, or the ability has not been switched on in configuration.
18008	Out of memory

Glossary

ACI (Autonomy Content Infrastructure)

The Autonomy Content Infrastructure is a technology layer that automates operations on unstructured information for cross enterprise applications, thus enabling an automated and compatible business-to-business, peer-to-peer infrastructure.

The ACI allows enterprise applications to understand and process content that exists in unstructured formats, such as e-mail, Web pages, office documents, and Lotus Notes.

APCM (Adaptive Probabilistic Concept Modelling)

Terms are given a weight according to their statistical importance in the DRE. Terms can have a weight between 0 and 255.

AXE (Autonomy XML Engine)

The Autonomy XML Engine (AXE) enables you to enhance the DRE's functionality by allowing you to input, output and process XML. It provides an infrastructure for complete automatic interoperability between applications using different tagging schemes, based on a conceptual understanding of XML documents, rather than on the tags themselves, and combines this with all other Autonomy functions.

Concept summary

A brief summary of each result document that is returned for a query. The concept summary displays a few sentences that are typical of the result's content (these sentences can be from different parts of the result document)

Connector

A Connector is an Autonomy fetching solution (for example HTTPFetch, Oracle Fetch, AutoIndexer and so on) that allows you to retrieve information from any type of local or remote repository (for example, a database or a web site). It imports the fetched documents into IDX or XML file format and indexes them into a DRE from where you can retrieve them (for example by sending queries to the DRE).

Context summary

Returns a conceptual summary of the result document that is biased by the terms in the query. A context summary comprises sentences that are particularly relevant to the terms in the query (these sentences can be from different parts of the result document).

Database

An Autonomy database is a DRE data pool that stores indexed information. The administrator can set up one or more databases, and specifies how data is fed to the databases.

DRE (Dynamic Reasoning Engine)

The Dynamic Reasoning Engine is a scalable, multithreaded process which is based on advanced pattern-matching technology that exploits high-performance probabilistic modeling techniques. The DRE contains databases into which you can index information using a Connector, the DRE Administration tool or an indexing command. You can then access this information through a front end or by sending query commands to the DRE.

IDX

Apart from XML files only files that are in IDX format can be indexed into a DRE. You can use a Connector to import files into this format or manually create IDX files (please refer to the chapter **Importing and indexing data into the DRE**).

Link term

Link terms (also referred to as "Links") are terms in query text that are also contained in the result documents that the DRE returns for this query.

Query

You can submit a natural language query to the DRE which analyzes the concept of the query and returns documents that are conceptually similar to the query. You can also submit Boolean, bracketed Boolean and keyword searches to the DRE.

Quick summary

A brief summary of each result document that is returned for a query. The quick summary displays the first few sentences of the result document.

Synonym file

A synonym file allow the DRE to handle synonym queries. A synonym query returns results which are conceptually similar to the query's terms and / or conceptually similar to the synonyms that are available for the query's terms. A synonym file contains comma separated lists of synonym strings for words. You can specify lists for each language type you have set up in the DRE within this file.

Index