Best Practices Guide

revision 4

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

Copyright © 2003 Autonomy.

Portal-in-a-Box, Classification Server, UAServer, HTTPFetch, AutoIndexer, Moreover Fetch, DIH, DAH, DiSH and DRE are trademarks of Autonomy.

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 Best Practices Guide.

Autonomy Solutions

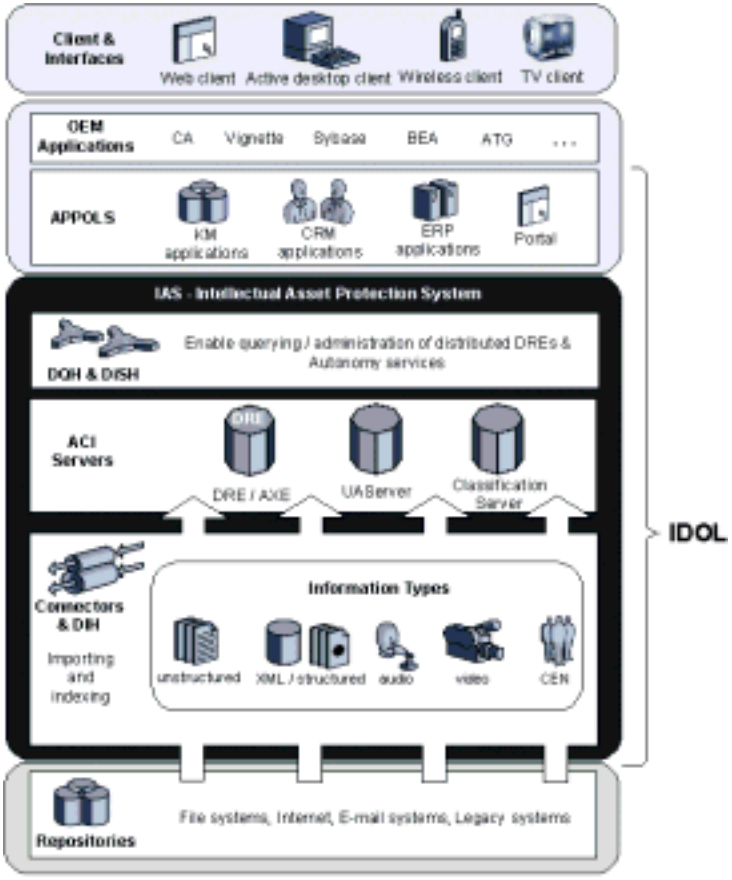
Autonomy solutions provide the software infrastructure that automates operations on unstructured information. This software infrastructure is based on IDOL server, the Intelligent Data Operating Layer. IDOL makes it possible for organizations to process digital content automatically and to enable applications 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

Autonomy provides the software infrastructure that automates operations on unstructured information. This software infrastructure is based on IDOL, the Intelligent Data Operating Layer. IDOL makes it possible for organizations to process digital content automatically and allow applications to communicate with each other. It consists of data operations that integrate information by understanding content, and is therefore data agnostic.



Information that you need in order to conduct business successfully comprises data in various repositories, for example, file systems (including multimedia files), the internet, e-mail systems or legacy systems.

Connectors

Autonomy Connectors (for example HTTPFetch, Oracle Fetch, AutoIndexer and so on) aggregate the information from its various repositories, import it to IDX or XML file format and index it into one of the ACI Servers. If you have multiple instances of an ACI Server installed, you can use the DIH (Distributed Index Handler) to distribute and load-balance indexing.

ACI Servers

The Autonomy ACI Servers (DRE 4 / AXE, UAServer, Classification Server) allow you to perform a variety of operations on different types of information.

The ACI (Autonomy Content Infrastructure) Client API is a technology layer that enables 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.

By automating operations on unstructured information for cross enterprise applications, the ACI API creates an automated and compatible business-to-business, peer-to-peer infrastructure. This allows enterprise applications to understand and process content that exists in unstructured formats, such as e-mail, Web pages, office documents, and Lotus Notes.

APPOLS and OEM applications

The information that has been indexed into and processed by the ACI servers can be accessed via various clients and Interfaces using APPOLS (Application Operating Layers, for example Portal-in-a-Box, Commerce Application Builder, Autonomy Answer, Voice Suite and so on) or OEM Applications (Portals such as IBM's WebSphere Portal, BEA's WebLogic Server and so on).

If APPOLS or OEM applications use multiple ACI Servers, **DQH** (Distributed Query Handler) and **DiSH** (Distributed Service Handler) make it possible to distribute and load-balance querying and service administration

IAS

The IAS (Intellectual Asset Protection System) provides integrated security to protect your data. At the front end, authentication checks that users are allowed to access the system on which result data is displayed. At the back end, entitlement checking and authentication combine to ensure that query results only comprise documents that the user is allowed to see, from repositories that the user is allowed to access.

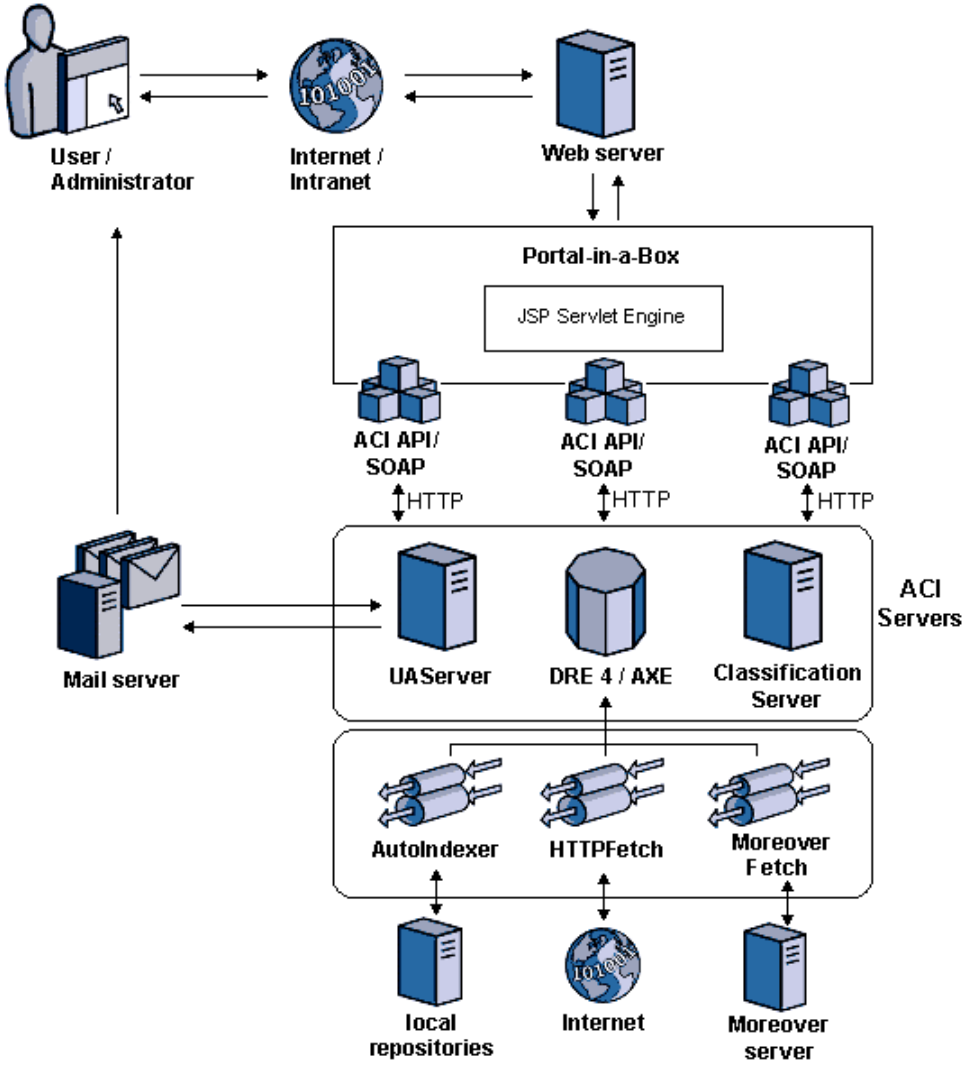
The following chapters provide you with hints for performing important stages of setting up or integrating your Autonomy software infrastructure. The information here is intended as an addition to the instructions in the documentation for each Autonomy component, to which you should refer for details of how to install and set up the product.

2. Planning, installing and configuring Portal-in-a-Box 4

Portal-in-a-Box is one of the most comprehensive Autonomy solutions and serves as a good example of how Autonomy products integrate with each other and with third party software. It comprises a number of core Autonomy modules and its functionality can be extended by integrating it with additional Autonomy modules. This versatility of the Portal, which enables it to be configured into a customized solution that fits perfectly into different environments, means that developers and system administrators who want to install the Portal need to consider the following questions:

- how should I distribute the Portal-in-a-Box components?
- do I need all the Portal-in-a-Box components?
- do I need additional components?
- what is the best way to integrate Portal-in-a-Box with pre-requisite third party components?
- in which order should I install components if I want to integrate Portal-in-a-Box with other Autonomy products?
- how much space do I need?
- how do I configure the components?

Typical Portal-in-a-Box set up



Users and administrators can access Portal-in-a-Box via their Intranet or via the Internet. The administrator can log on to Portal-in-a-Box in order to access the Portal's user and administration features.

The Portal-in-a-Box interface that users and administrators access is displayed in a web browser. If a user or the administrator makes a request to the web server, it passes it to the JSP servlet engine which processes the request and then passes the HTML that it has generated back to the web server.

The Autonomy ACI API is the interface between the Portal and the ACI servers from which it retrieves information. It is written in Java.

The Autonomy Connectors AutoIndexer, HTTPFetch and Moreover Fetch aggregate data from local sources, the Internet and the Moreover Server. They import the data to IDX format and then index it into the Portal-in-a-Box Content DRE’s databases.

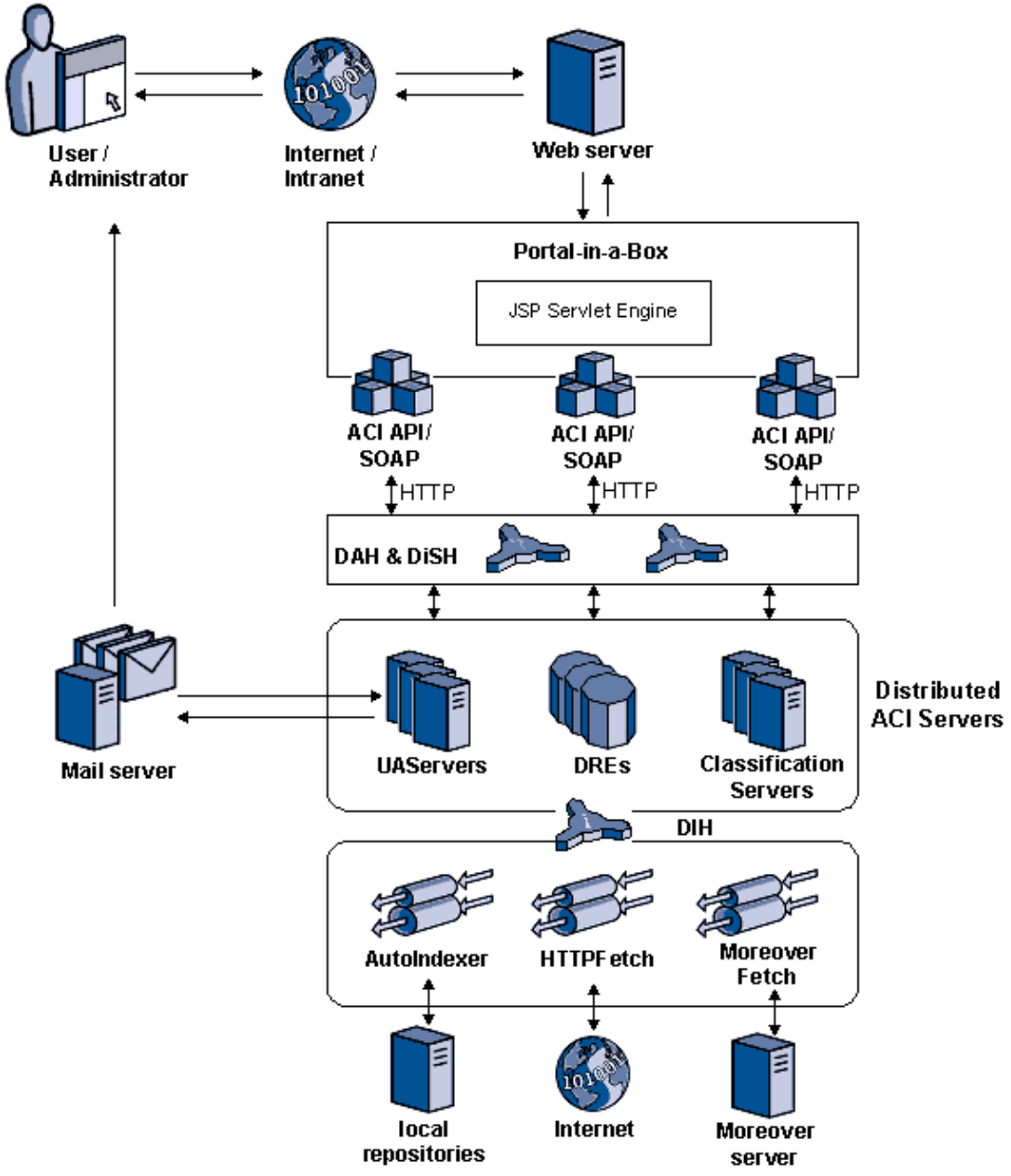
UAServer stores user details, roles, agents and profile agents. It indexes user agents and profile agents into an Agent DRE in order to make them available for Community queries. It also stores security data (user permissions and authentication details) that it aggregates from any group servers that it is connected to. This ensures that existing security policy is maintained.

The UAServer's Mailer functionality requests a list of all users and their agents from the UAServer (which stores the agents of Autonomy front end users), and then periodically requests the UAServer to query the DRE with the User's agents.

UAServer sends the agent results that it receives from the DRE to Mailer, which then mails the results to the Portal users via a Mail server. By default Mailer only sends new agent results to users, not results that have been sent before.

Classification Server identifies concepts in the data in a DRE, and uses them to build clusters of related information. Taxonomy generation builds a hierarchical structure, either from these clusters or from the results of a query to the DRE, which you can write to disk as a directory structure or import into the category hierarchy. The category hierarchy contains categories that Classification Server builds from concepts identified in your training material or imported by taxonomy generation.

Portal-in-a-Box setup in a distributed environment



Users and administrators can access Portal-in-a-Box via their Intranet or via the Internet. The administrator can log on to Portal-in-a-Box in order to access the Portal's user and administration features.

The Portal-in-a-Box interface that users and administrators access is displayed in a web browser. If a user or the administrator makes a request to the web server, it passes it to the JSP servlet engine which processes the request and then passes the HTML that it has generated back to the web server.

The Autonomy ACI API is the interface between the Portal and the ACI servers from which it retrieves information. It is written in Java.

The DAH distributes ACI action commands to DRE 4 / AXE ACI servers, increasing the speed with which actions are executed and saving processing time. Multiple copies of DREs to which the DAH distributes actions ensure uninterrupted service if any of the DREs should fail.

The DiSH allows you to manage the Autonomy applications that the Portal-in-a-Box set up comprises from one central point.

The DIH allows you to distribute index commands to DRE 4 / AXE ACI servers, so that index commands are executed more quickly and processing time is saved. Multiple copies of DREs to which the Distributed Index Handler distributes index commands ensure uninterrupted service if any of the DREs should fail.

The Autonomy Connectors AutoIndexer, HTTPFetch and Moreover Fetch aggregate data from local sources, the Internet and the Moreover Server. They import the data to IDX format and then index it into the Portal-in-a-Box Content DRE’s databases.

UAServer stores user details, roles, agents and profile agents. It indexes user agents and profile agents into an Agent DRE in order to make them available for Community queries. It also stores security data (user permissions and authentication details) that it aggregates from any group servers that it is connected to. This ensures that existing security policy is maintained.

The UAServer's Mailer functionality requests a list of all users and their agents from the UAServer (which stores the agents of Autonomy front end users), and then periodically requests the UAServer to query the DRE with the User's agents.

UAServer sends the agent results that it receives from the DRE to Mailer, which then mails the results to the Portal users via a Mail server. By default Mailer only sends new agent results to users, not results that have been sent before.

Classification Server identifies concepts in the data in a DRE, and uses them to build clusters of related information. Taxonomy generation builds a hierarchical structure, either from these clusters or from the results of a query to the DRE, which you can write to disk as a directory structure or import into the category hierarchy. The category hierarchy contains categories that Classification Server builds from concepts identified in your training material or imported by taxonomy generation.

Planning

Step 1: determining project needs

Before you select the Portal-in-a-Box components, you need to determine what the needs of your project are, for example:

- how much data do you want to process?
- what do you want to do with your data?
- what repositories store the data that you want to process?
- how many people are going to use the Portal?

Step 2: selecting Portal components

Decide which of the Portal-in-a-Box components you want to install. The following are available:

Content DRE (fully licensed)

An Autonomy DRE 4 into which data is indexed by HTTPFetch, Moreover Fetch and AutoIndexer. The Portal-in-a-Box front end communicates with the Content DRE through HTTP commands.

Unless you want to integrate the Portal with a DRE that you have already set up, you always need to install the DRE. If you already have a DRE that you want to integrate the Portal with, you can install the Portal DRE in addition, for example if you want a backup DRE or spread your data across multiple DREs.

Please refer to **Step 3** for details on how to determine if you require multiple DRE installations

UserAgent Server (fully licensed, includes the Mailer)

The UAServer allows Portal-in-a-Box to handle requests for user and agent details. It stores a database of users and their agents and profiles centrally so that many applications can access the user/agent information, provided they are connected via HTTP to the computer running UAServer.

If the repositories from which data is indexed into the Content DRE are connected to group servers, you can configure UAServer to read the security information that the group servers store, and use it to determine which data users can access via the Portal.

The UAServer's Mailer functionality allows you to set up a process that automatically mails users their agent results.

Unless you want to integrate the Portal with a UAServer that you have already set up, you always need to install the UAServer.

Classification Server (partly licensed)

The Classification Server provides the following functionality:

Categories

Classification Server identifies concepts in training material such as documents, topics from a legacy keyword engine or Autonomy Agents, and uses these concepts to build categories. This allows you to query Classification Server in order to find documents or categories that match your category or to find categories that match a document you specify.

Clusters (requires an additional license)

Starting from a snapshot you can automatically categorize data within that snapshot so that related information is clustered together (this does not require the setup of an initial taxonomy). Each cluster represents a concept area that contains a set of items that share common properties. Clustering data allows you to make trends and developments in data visible.

Taxonomy (requires an additional license)

The Classification Server's taxonomy generation feature allows you to create automatically hierarchical contextual taxonomies of clusters or other information. This provides you with an overview of the 'information' landscape and an insight into specific areas of the information.

You need to acquire an additional license for Taxonomy if you want to be able to automatically create categories from taxonomies. You can then display and modify these categories within the following portlets:

Channels

The Channels portlet allows you to display categorized information that is relevant to activities and interests within your organization.

Legacy Content Manager

The Legacy Content Manager portlet allows you to import legacy topic sets to categories, and to administer the categories that users can access in the Channels portlet.

You need to acquire an additional license for Clustering if you want to be able to use the following portlets:

BreakingNews

The BreakingNews portlet allows you to display the latest information for particular subjects.

HotNews

The HotNews portlet allows you to display the most relevant information that is available for particular subjects.

Spectrograph

The Spectrograph portlet displays a spectrographic view of data that stems from a particular period in time.

2D Map

The 2D Map portlet allows you to view the strength and spread of subjects in a cluster of related topics.

HTTPFetch (fully licensed)

HTTPFetch is an Autonomy Connector which allows you to download documents from web sites and index them into the Content DRE.

You should install the HTTPFetch if you want to aggregate documents from http sites (this includes Domino.Doc and QuickPlace). Note that if you want to aggregate documents from a local file repository that you can access via http, you should use AutoIndexer rather than the HTTPFetch (AutoIndexer can download the documents faster while http connections usually have a restricted bandwidth).

AutoIndexer (fully licensed)

AutoIndexer is an Autonomy Connector which automatically checks specified directories for new documents and indexes them into the Content DRE.

You should install AutoIndexer if you want to aggregate documents from any files that are stored on disk.

Moreover Fetch (fully licensed)

Moreover Fetch is an Autonomy Connector which allows you to download documents from the moreover.com web site and index them into the Content DRE.

You should install Moreover Fetch if you want to aggregate news documents (world news, business news and so on).

JSP front end interface & Autonomy ACI API (fully licensed)

The Portal-in-a-Box user and administrative user interface. The user interface allows users to use the Autonomy portlets to access the data that is stored in the Content DRE (depending on their permissions). The administrative user interface allows administrators to manage the user interface.

Step 3: selecting additional components

Determine if you need additional products.

DREs

The number of DRE installations you require generally depends on the number of documents that you want the DRE to handle. Generally one DRE can comfortably store up to 10 million documents. To achieve optimal query speed (or if you want to use the DRE for search engine type queries), you should store no more than 2 million documents in a DRE.

It is also useful to install multiple DREs for the following reasons:

- To improve query speed. For example, if your data exceeds 2 million documents, and can be separated into broad categories, the DRE response time is minimized if you spread the data across multiple DREs, so that each DRE stores one subject area.

  Similarly, if you have data that can be separated into data that is used frequently and data that is used infrequently, you should store each data group on individual DREs to speed up the DRE's response time.

- To set up fail-over and load balancing in connection with a DAH or DIH installation. Please refer to the appropriate manual for further details.

- To create a backup of the DRE. This provides you with a safe copy of the DRE and the data it stores.

Note: if you want to index less than 2 million documents into a DRE, and these documents can be separated into different topics, you should create a DRE database for each of the topics in order to maximize the DRE's response time.

Connectors

The Connectors you require generally depend on the data that you want to aggregate. If you want to retrieve documents from an Oracle or ODBC database, for example, you need Oracle or ODBC Fetch.

Another important consideration when deciding which Connectors to use is how quickly content that you want to index into the DRE needs to be aggregated. For example:

- A news feed that can be accessed via http or disk would be aggregated more quickly by AutoIndexer than HTTPFetch. Similarly, if you want to aggregate an intranet site, AutoIndexer is a better solution than HTTPFetch.

- Domino.Doc and QuickPlace documents can be aggregated using the HTTPFetch but will be handled more efficiently by Notes Fetch.

- Complex Oracle databases, which would require much time configuring, could be better indexed via XML output.

DIH

If you want to index documents into multiple DREs using fail-over and load balancing, you need to install a DIH. This enables you to scale your system in a linear manner, distributing the execution of your index commands and saving overall processing time.

The DIH also enables you to administer multiple DREs simultaneously.

DAH

If you want to distribute ACI action commands between multiple DREs using fail-over and load balancing, you need to install a DAH. This enables you to scale your system in a linear manner, increasing the speed with which actions are executed and saving processing time.

DiSH

DiSH is a useful product to install, if you want to automatically send service commands to the multiple Autonomy applications that your Portal installation comprises in order to control their operation and monitor their performance. It also allows you to view and modify the configuration of these applications and to generate statistics for them.

Step 4: distributing components

Determine where you want to install the individual components. Ideally you should distribute them across multiple servers as follows:

Machine 1:	• **Portal-in-a-Box front end** (Portal-in-a-Box application, Application server, Web server)	
Machine 2:	• **DRE 4 / AXE**	
Machine 3:	• **UAServer**	
Machine 4:	• **Classification Server**	
Machine 5:	• **DAH** • **DiSH** • **DIH**	• **AutoIndexer** • **HTTPFetch** • **MoreoverFetch**

Note: while you may not be able to spread out the individual components like this, you should avoid installing the DRE on a machine that hosts Connectors (as importing can be very system resource intensive) or another DRE (to avoid two DREs competing for disk access).

Step 5: checking pre-requisite software

Check if you have prerequisite software installed:

A J2EE 1.2 compliant servlet engine that runs in conjunction with a web server

If you have problems integrating your web server with Portal-in-a-Box, it is recommended that you install Resin which can easily be integrated with the Portal (please refer to the **Installation** section for details).

Note:

- If you want to use a separate web server and servlet engine, your web server must be configured to use the servlet engine to serve web applications.
- Before you start installing Portal-in-a-Box, you must determine your web server's web publishing directory and the servlet engine's default web application directory
- If NT automatic login is required the servlet engine needs to be installed in conjunction with IIS.

Step 6: determining security requirements

Decide if you need to set up security, and, if so, what level of security you require:

- **Front end security**

  Authentication to check that users are allowed to access the system on which result data is displayed.

  Note that if you require NT automatic login, your servlet engine needs to be installed in conjunction with Microsoft IIS.

- **Back end security**

  Entitlement checking and authentication combine to ensure that query results only comprise documents that a user is allowed to see.

Please refer to the IAS manual for details on setting up security.

Installation

The following provides you with an overview of the steps that the installation of Portal-in-a-Box with additional modules requires. If you do not want to install additional modules, you should ignore the steps that apply to them and only concentrate on the Portal-in-a-Box specific steps.

Note: due to the variety of complex set ups that different servlet engines require, the integration of Portal-in-a-Box with servlet engines and web servers can be work intensive. It is therefore recommended that you integrate the Portal with a Resin servlet engine which contains a web server and can easily be installed with the Portal. The following installation overview includes the steps a Resin installation would require.

1. Download Resin 2.1.x as a zip from http://www.caucho.com/ and unzip Resin file to a directory (for example, <resin_home>).
2. If required, install Resin as a service.

   For example install Resin as a Windows Service, run the following from a command line:

   <resin-home>\bin\httpd -install

   This will create a Windows Service with the default name of the Resin web server.
3. Run the Portal-in-a-Box installer. Only install the components you require.

   When the **Web Publishing Directory** dialog is displayed, select **<resin_home>/webapps** as the web publishing directory and **8080** as **Your web server port** (make sure that no other process is using this port). When the **Setup Servlet Engine** dialog is displayed, configure Resin as your servlet engine.

   Please refer to the installation instructions in the Portal-in-a-Box manual for details on the installation.
4. Start Resin (either through the Services Management Console if you have installed it as a service or by running **<resin-home>\bin\httpd** from the command line. Running **<resin-home>\bin\httpd -verbose** is a good way of getting diagnostic information from Resin).
5. Start the Portal services and check that the Portal loads successfully by clicking on the **Portal front end** or **Portal Administration** link . Ensure that the services and the servlet engine are working, and the site is visible to the appropriate parts of the network.
6. Install any additional DREs or Connectors that you require.
7. If you want to integrate Portal-in-a-Box with them, install DIH, DAH or DiSH. Please refer to the installation instructions in the appropriate module's manual.

   Note that the DIH installation requires you to specify the IP address of one of the DREs into which it will index data, and the DAH installation requires you to specify the IP address of two of the DREs to which it will distribute actions. You can use the configuration file of the DIH and DAH to connect to additional DREs (please refer to the appropriate manual).

Configuration

Step 1: DiSH configuration

If you require the DiSH, DIH and DAH you should consider the following points when you configure them:

DiSH

If you require the DiSH, you should configure it first. Ensure that the DiSH dashboard is working with the servlet engine that you have integrated with Portal-in-a-Box (please refer to the DiSH manual for details). It is recommended that you use the dashboard to configure the DiSH.

DIH

If you require the DIH, you need to decide which mode you want to run it in:

- **Mirror**

  The DIH distributes all the index data it receives to all the DREs that it is connected to. The DREs are exact copies of each other which must all be configured in the same way.

  You should run the DIH in Mirror mode if you want to ensure uninterrupted service if one of the DREs should fail. While one DRE is inoperable, data continues to be indexed into its identical copies which at the same time are still available to return data for queries.

- **Non-Mirror**

  The DIH distributes the index data it receives evenly across the DREs that it is connected to. For example, if the DIH is connected to 4 DREs, it indexes approximately one quarter of the data into each one of the DREs (individual documents are not split up).

  You should run the DIH in Non-mirror mode if the amount of data that you want to index is too large for a single DRE. If the DREs that the Distributed Index Handler indexes into are situated on different machines, the index process will require less time.

DAH

If you require the DAH, you need to decide which mode you want to run it in:

- **Mirror**

  The DREs that the DAH distributes ACI actions to are identical (that is all of the DREs are exact copies of each other, each one is configured the same way and contains the same data).

- **Non-mirror**

  The DREs that the DAH distributes ACI actions to are different (that is each DRE is configured differently and contains different data). If you are running the DAH in Non-mirror mode, you need to set up Virtual Databases that can be of the following types:

  Combinator

  The Virtual Database forwards an action command to all the databases that it comprises. It collates and sorts the results before it returns them.

  Distributor

The Virtual Database forwards an action command to one of the databases it comprises. These databases must be identical (that is all of the databases are exact copies of each other and contain the same data). The way it forwards the action is determined by the distribution method.

Step 2: DRE configuration

Consider the following points for each Data or Content DRE you are setting up:

Default configuration

You should always review the default configuration. While the DRE will work "out-of-the-box", its default configuration may not suit your personal environment. You need to optimize its performance within your environment, and ensure that it will work smoothly with the components that you are integrating it with.

Index file size

If you are indexing many small IDX files (files that contain less than 2K of text), you should enable **DelayedSync**, so that the DRE stores index data every time it finishes indexing an IDX or XML file or when a specified interval has elapsed.

Index fields

Decide which fields you want to use as index fields. Identifying index fields allows you to optimize the query process when you restrict queries using these fields. This means that index fields should hold data that you are likely to use frequently in order to restrict queries. A typical example is to restrict the results returned by a query by document type. The extension of each document can be extracted by the Connectors and added to a */FILEEXTENSION field.

While you should generally avoid querying non-indexed fields for performance reasons, this is preferable to indexing 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.

Preventing duplicate content

Decide which fields you want to use as reference fields (for example, */DRETITLE), and set up a field process that identifies these fields as reference fields when a Connector indexes documents into the DRE.

Configure the DRE to eliminate duplicate copies of documents at index time by setting the DRE's **KillDuplicates** parameter to one or more of the reference fields that you have defined .

DRE databases

Configure the databases that you want the DRE to contain. You should give the databases names that are indicative of the content that they will be storing. For example, you could create a Marketing database to store data that will be indexed from a Marketing network.

You will need the names of the databases when you configure the Connectors in order to determine into which of the DRE databases the Connectors should index.

Security

If you require security, you should set up a field process that detects the security type of documents, and configure settings that define how individual security types should be treated.

Languages

If you want to index languages other than English into the DRE, you need to use automatic language detection or a field process that determines the language of documents:

- **Automatic language detection** (requires an additional license)

  Allows the DRE to automatically detect the language and encoding of documents when they are indexed. This is useful if you want to index a very large amount of data that can comprise multiple languages and encodings (for example, if you want to index company data that stems from international branches of the company, if you want to index Russian documents that use different encodings or if you use the HTTPFetch to spider data that could use multiple languages and encodings).

- **A field process that determines the language and encoding of documents**

  You can set up a field process that identifies the language types of documents using specific fields. This is useful if you are indexing documents that contain metadata which identifies their language and encoding, as this guarantees the identification of the language type to be accurate (due to the nature of data, a negligible number of documents may not identified by automatic language detection).

You should also review the following language configuration settings as it is essential that they are configured according to your requirements:

DefaultLanguageType

ProperNames

IndexNumbers

Note: if your project requires exact matches for search, you can disable **Stemming**. However, it is generally not recommended to disable **Stemming** as this may have a negative impact on query performance and for some languages decrease the relevance of results (when conceptual queries are performed)

Step 3: Connector configuration

The configuration of the Connectors is the most important element in making the system perform optimally. Consider the following points carefully for each Connector you are setting up:

DIH

If you have installed a DIH, you need to ensure that the Connectors index into the DIH instead of the DRE.

Run time and intervals

Consider how much time the Connectors will require to extract data, the DRE volume the data will require and how long the indexing of the data will take. If you are using multiple Connectors that are located on the same machine, you should ideally configure them to run when other processes are not system intensive (for example, at night), and ensure that they don't all run at the same time.

Make sure the Connectors run in sensible intervals, and that the requests they make are appropriate for the source from which they are aggregating (for example, if you use HTTPFetch, you should ensure that it does not send too many requests to a site, as it may otherwise be blocked as a malicious spider).

Security

If you want to use mapped security, you need to configure the Connectors to retrieve the security information of documents. The document ACLs are then encrypted and mapped to a DRE field.

Note that for the AutoIndexer you can install the ACLCheck security plug-in which allows you to check if documents' ACLs change and update them accordingly in the DRE. The ACLCheck security plug-in is available on request.

Quality of content

You should only index content that is useful to the intended purpose for the Autonomy application. Identify which parts of document content you want to use (for example what field or metadata a Connector should index as a document's title), and which parts you don't want to use. Some common considerations are:

- Import useful information about documents into fields. For example, if you want information about the size and the type of documents that are indexed into the DRE to be available, you could specify the following import settings in your Connector's configuration file:

  ImportExtractLength=1
  ImportExtractLengthToField=FILESIZE

  ImportExtractExtension=true
  ImportExtensionToField=FILEEXTENSION

- Remove repetitive strings within the content. If a document, for example, contains multiple copyright strings (©Copyright 2003 Autonomy Corporation), there is no reason to index all of them. You can usually remove such elements by defining the start point and end point of the text that you want to index.
- Discard Excel documents. Unless you are dealing with XLS documents that contain rich textual content or the indexing of numeric data has a specific use for you, it is counterproductive to set the Connectors to import these files.
- Restrict the size of documents that the Connectors are permitted to process. The import process can waste a lot of time processing excessively large documents that may be just images. Use **MaxSize** to set a maximum size for documents that are processed by the Connectors (this can also be useful for handling XLS documents).

Synonyms

If your project involves searching for product or company names, you should consider enabling synonym querying in the DRE and creating a synonym file that lists the required product or company names alongside common misspellings.

Content distribution

Determine into which DRE databases the Connectors should index the content that they aggregate. You can divide content between DREs and their databases according to the following methods:

- Allocate a DRE database to each content store of the source repository. You can, for example, set up a DRE database per Notes database, and then restrict users to only be able to query this particular source. This would entail a maintenance overhead but also allow administrators to view the exact number of documents that is available for each source.
- Group the aggregated content into thematic databases. You can, for example, store all Exchange and Notes content in an “internal email” database.

Preventing duplicate content

If you have configured the DRE to remove duplicate content, the DRE's method of deduplication is used by default. You can, however, configure your Connectors to override the DRE's deduplication method:

- If you are only indexing IDX files, a document's DREREFERENCE is usually used to identify and remove duplicate content. This, however, is often not adequate if identical content has been duplicated across the content sources. If this is the case, it is practical to create a unique reference field value by combining various fields (for example, a DUPLICATES field that combines a DRETITLE, FILESIZE and FILEEXTENSION field), and configuring the Connectors to use this field to identify and remove duplicate content. Use the Import module's FieldGlue settings to create a unique field (the Import module is included in each one of your Connectors).
- You should avoid removing duplicate documents using the REFERENCEMATCHnn option, if it is imperative to index as much of the content as possible. REFERENCEMATCHnn identifies duplicate concepts rather than content, and is likely to discard a great deal of good quality content if you set it to a low value.

PDF files

If you are using your Connectors to import PDF files, you should note that it is possible to both highlight matching link terms within a document and link to the relevant section of the PDF. This highlighting functionality is available on request.

Notes Fetch

If you are integrating your Notes Fetch with your Portal-in-a-Box installation, you should note that by default this Connector maps all the metadata that is stored with a Notes document to fields of the same name in the DRE. As DRE4 stores all fields this can be costly in terms of space usage. However, you can avoid this problem by doing one of the following:

- Set **MapAll** to **false** to prevent the Notes metadata from being mapped to equivalent DRE fields, and map only Notes fields you require to DRE fields (when configuring individual fetch jobs).
- Define **CantHaveCSVs** in the DRE configuration file to prevent the DRE from storing fields that are not required.

Step 4: Test index

Configure AutoIndexer to aggregate from a specific location on disk, and store about 10 files of various types (PDF, XLS, DOC, XML and so on) in this location. Start AutoIndexer and check the indexing of the files by sending an **IndexerGetStatus** command and some test queries to the DRE. Check if AutoIndexer has treated the different file types as you have specified, and if all the metadata that you require is available.

Step 5: Adjusting the DRE configuration

Make any required adjustments to the configuration of your Content or Data DREs.

Step 6: Adjusting the Connector configuration

Make any required adjustments to the configuration of your Connectors.

Step 7: Index

Start your Connectors to index your data into the DRE. Make sure you always use as much cache as possible, but be careful to reduce the index cache and increase the query cache once the main indexing job has been completed. Do not set the caches to more than about 75% of the available RAM (when added together) as you may run out of RAM and cause the DRE to fail.

Step 8: Classification Server configuration

Configure the Classification Server. Please consider the points addressed in the chapter **Clustering with Classification Server** before you start the configuration.

Step 9: Front end customization

The customization of the front end determines the Portal's user-friendliness. In order to ensure that the Portal meets the needs of your environment, you should consider the following points:

Functionality display

Decide which elements of the product are most useful for the end users. You should configure the front end to display particularly useful features by default, so they can easily form a part of the users' general work flow.

Useful information

If you have identified information in your data that users will generally require and configured your Connectors to import this information, you should consider if you want to display some or all of it alongside results by default. You could, for example, modify the Portal JSPs to display the author, creation date, file type and file size of a document when this document is displayed as a result.

Step 10: UAServer configuration

Configure the UAServer in accordance to the security setup you require:

If you are using a version 4 security type you must set **CaseSensitiveUsernames** and **CaseSensitiveGroupNames** to **false** in the configuration file section that sets up the security type.

(A version 4 security type is one for which the value that you specify with the Type parameter in the DRE configuration file section that sets up the security type contains the string "V4". For example, Type=AUTONOMY_SECURITY_V4_NT_MAPPED).

Step 11: Setting up users

You can create a user structure for Portal-in-a-Box in a number of ways. You should consider the following points in order to set up a user structure as efficiently as possible:

Creating a user structure from scratch

Before you create users, you need to set up roles and define their permissions. Once you have done that you can create users from scratch using one of the following methods:

- You can create each user yourself from the Portal's administrative pages, and allocate this user to one or more roles.
- You can give users passwords that enable them to register as a Portal user and allocate them to appropriate roles.

Importing a third party user structure

You can use the UAServer to import an existing third party user structure into the Portal (using the **DeferLogin** option):

- If the user structure that you are importing is connected to a group server (for example, Notes, NT or Documentum), you can use the UAServer's SyncRolesFromGroups option to automatically create a role for each legacy group.
- If you are importing an LDAP user structure, you can configure the UAServer to import the users' details into user fields (using **IndexFieldCSVs**).

Default user creation

Portal-in-a-Box creates a default user for you. The default user role is given to any user who uses the Portal without registering, and sets how much of the Portal's functionality a user who has not registered can access. This means that you need to customize the default user in order to assure appropriate access rights apply to him.

Template user creation

You should create a template user for each of the roles that you have set up within your Portal. A template user serves as a template for a User interface that is specific to a particular role. You can thus create a different User interface for each role, which is displayed when a user logs in to the Portal for the first time (before he customizes the Portal).

Step 12: Testing users

Before your Portal becomes live, it is recommended that you log in to the Portal using the different template users (which your administrative status allows you to access) and test if you have set up the roles permissions correctly.

You should also permit a small number of users to log in to the Portal and test its setup. This provides you with feedback that allows you to ensure that the Portal is optimized for the different roles.

3. Modifying configuration files

Modifying configuration parameter values

Applying modifications to configuration files

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

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 escape 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

4. Extracting document titles

Informative document titles reduce the amount of time that the user spends sorting through sets of results. A good title should uniquely identify a document and give some idea of its content. Meaningless titles, such as "CNN: News" or "doc123456", can result in time wastage and extra cost, as the user has to look beyond the title for each document in order to find out anything about its content.

Document titles are stored in the **DRETITLE** field when documents are indexed into the DRE. By default, the "Title" field of the document is extracted for this purpose, but you can configure the Import module and Connectors in order to change the title that the DRE stores in one of the following ways:

- Generate a title from a document's content
- Manipulate a title field before extracting a title from it
- Specify a field to extract titles from

Generating titles from a document's content

The following Import module settings in you Connector's configuration file allow you to change the way titles are extracted from documents.

Note: if you are generating titles from a document's content, it is important that you set the Import module to import good quality content. See the **Content** chapter for details of how to do this.

ImportIntelligentTitleSummary

Enter **true** to generate a unique intelligent title and summary for the documents that are imported into the DRE. An intelligent title and summary are generated by conceptually comparing each document title and summary with the next. If the document title is in use already as another document's title field, then the import module extracts a unique title from the conceptual content of the document.

The default value for this setting is **false**.

ImportMinTitleChars

Allows you to specify the minimum number of characters that can be used as a title for imported documents. If a title of less than the specified number of characters is found, the title is taken from the document content instead. By default, **ImportMinTitleChars** is set at **3**.

ImportTitleSkipWords

Enter the number of words that you want to skip before obtaining a title from the document. By default this is **0**.

ImportSectionTitles

Enter **true** to import each document section with a different title taken from the anchor. Enter **false** to turn off the titles for each section of a document, so that each section is imported with the same title. This is useful when spidering E-commerce sites that contain an anchor for each product. Disabling the individual section titles means that you can use Autonomy's Distributed Query Handler (DQH) to duplicate by title later on (since each review will have the same title).

The default value for this setting is **false**.

Manipulating title fields before extracting a title

If your documents' titles regularly contain useless characters or strings, you can set import field processing parameters in your Connector's configuration file in order to remove those characters before the Import module generates titles.

Please refer to the Import module documentation for details of these settings.

Specifying a field to extract a title from

A document's title field may not always be the best from which to set the **DRETITLE** field (for example, if there are a number of documents with the same title). You can specify another field that you want to map to the **DRETITLE** field with the following settings in the Import module's configuration file:

ImportRemapField*n*

Specify the fields whose value you want to remap to the value of the corresponding **ImportRemapFieldTo*n*** fields.

ImportRemapFieldTo*n*

Specify the fields in which you want to store the values of the corresponding **ImportRemapField*n*** fields.

For example:

ImportRemapField0=Description

ImportRemapFieldTo0=*/DRETITLE

In this example, when a document is imported, the contents of a document's **Description** field are saved as the **DRETITLE** field for the document.

HTML documents

The content of the <TITLE> field of an HTML file doesn't necessarily reflect the content of the HTML page. You can specify any of the other fields contained in the HTML with **ImportRemapField*n***.

Non-HTML documents

You can specify any of the fields contained in your documents with **ImportRemapField*n***.

If you want to view the fields that the Import module slave for your documents' format has extracted from the document, use the following command:

On NT: <slave>.exe <file_name1>.<file_extension> <file_name2>.html

On UNIX: <slave>.exe -cmd convert <file_name1>.<file_extension> <file_name2>.html

For example, using PDF slave:

On NT: pdfslave.exe <file_name1>.pdf <file_name2>.html

On UNIX: pdfslave.exe -cmd convert <file_name1>.pdf <file_name2>.html

<file_name1>

Enter the name of a PDF file for which you want to find out field names.

<file_name2>

Enter the name of the HTML file to which you want to write the results of this command.

The HTML file that is produced contains, for example:

<META NAME="CreationDate" CONTENT="D:20010419185953">

<META NAME="Producer" CONTENT="Acrobat Distiller 4.0 for Windows">

<META NAME="Title" CONTENT="042001_TVLY_SMF_Note">

<META NAME="Keywords" CONTENT="TVLY">

<META NAME="ModificationDate" CONTENT="D:19990912205731">

<META NAME="Subject" CONTENT="pdfmark 2.0">

<META NAME="MyKey" CONTENT="HQ Document Publishing">

<META NAME="Creator" CONTENT="Hand Programmed">

<META NAME="Author" CONTENT="Steve Fitzgibbons">

<META NAME="ModDate" CONTENT="D:20010419185954-07'00'">

Lotus Notes

Lotus Notes is a very meta-data-rich content source as it stores all documents within a structured format, which is determined by the design of the database. The title field of Notes documents is unlikely to be a useful document title. You can set another field with the following setting in the Notes Fetch configuration file:

NotesTitleField

Specify the name of the field in the Notes database that is mapped to the **DRETITLE** field in the DRE.

For example:

NotesTitleField=Author

The best approach for determining what fields are contained in your Notes database is to fetch a sample set of the content and allow the Connector to create the IDX file but not index this into the DRE. View the structured IDX file in order to see which field is best suited as the document title. Please refer to the **Create structured IDX files without indexing documents into the DRE** section in the **Configuring data storage** chapter for details of how to create the IDX file for this purpose.

5. Generating document summaries

Good summaries of result documents enable the user to tell at a glance which documents are most useful to them. Autonomy has the ability to generate real-time conceptually relevant summaries that can be influenced by why the user asked the question.

The quality of summaries depends on the content that you have imported and indexed. Please refer to the **Good quality content** chapter for details of how to optimize the quality of your content.

By default, no summaries are produced, but you can instruct the Import module and the DRE to generate summaries automatically during indexing and when queries are sent to the DRE. Unless you specify otherwise, these summaries are stored in a document's **Summary** field. The most appropriate time to generate summaries depends on the kind of summaries you expect to use:

Summary generated during:	**Usage:**	**Where to make settings:**
importing / indexing	Summaries need to be generated just once - for example because you expect the same kind of summary to be required for all result documents	• Import module settings in your Connector configuration file (DRE 4) • DRE configuration file (DRE 3 only)
querying	Summaries need to be generated frequently - for example because you expect different queries require different kinds of summaries (context/content/quick)	• DRE 4 configuration file, • query string sent to DRE

Generating summaries during importing / indexing

Generating summaries during importing with the Import module

If your DRE version is DRE 4, and you want to include summaries when documents are indexed into the DRE, you can set the following Import module parameters in your Connector's configuration file:

ImportSummary

Enter **true** to import the first sentences of documents as a summary into the DRE. You can specify the number of sentences using **ImportSummarySize**.
Enter **false** if you don't require a summary. This is the default.

ImportSummarySize

If you have set **ImportSummary** to **true**, you can specify the number of sentences that you want to import into the DRE as a document summary. The sentences are taken from the start of the document. The default value for this setting is **3**.

ImportIntelligentTitleSummary

Enter **true** to generate a unique intelligent title and summary for the documents that are imported into the DRE. During the importing process titles and summaries that are repetitive across an import branch are deleted.

By default this is set as **false**.

Generating summaries during indexing with DRE 4 / AXE

DRE 4 / AXE does not allow you to generate document summaries during indexing. You can use summaries generated by the Import module, or generate summaries when you send queries to the DRE.

Generating summaries during indexing with DRE 3

DRE 3 allows you to generate summaries at index time. The following settings in the DRE 3 configuration file's [Server] section allow you to set the kind of summaries that are generated:

SummaryOverride

Enter **true** to enable the DRE to overwrite summary fields with its own summary. This is the default.

[IndexSummary] section

Method

Allows you to specify the kind of summary you want to create. You can set **Method** with the following values:

CONCEPT

Generates a conceptual summary of the document. A concept summary comprises sentences that are typical of the document's content (these sentences can be from different parts of the document).

QUICK

Generates a quick summary of the document. A quick summary comprises the first few sentences of the document.

NONE

No summary is generated. This is the default.

MinWords

If you have set **Method** to **QUICK**, enter the minimum number of words that the quick summary should consist of. By default this is **0**.

MaxWords

If you have set **Method** to **QUICK**, enter the maximum number of words that the quick summary should consist of. By default this is **0**.

Sentences

If you have set **Method** to **CONCEPT**, enter the number of sentences that the concept summary should have. By default this is **0**.

FieldNum

Enter the number of the field in which you want to store the summary.

Configuration file example

The following shows an example of a section of the DRE configuration file:

```
[Server]
SummaryOverride=true

[IndexSummary]
Method=Concept
Sentences=4
FieldNum=3
```

Generating summaries during querying

Generating summaries during querying with DRE 4

You can instruct the DRE to return summaries along with results for the following actions:

- Query
- Suggest
- SuggestOnText
- Summarize

Summaries are generated for each result document during the query. Include the following parameters in the query/command string for these actions in order to specify the kind of summary and length of summary that you want:

- Summary=<summary_type>
- Sentences=<length_of_summary>

For example:

http://<IP_address>:<aci_port>/action=query&Text=<query_text>&Summary=<summary_type>&Sentences=3

Summary=<summary_type>

You can set the following values for **<summary_type>**:

Concept

Returns a conceptual summary of the result document. A concept summary comprises sentences that are typical of the result's content (these sentences can be from different parts of the result document).

Context

Returns a conceptual summary of the result document that is biased by the terms in the query **Text** and/or **FieldText**. 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).

Quick

Returns a brief summary of the result document. A quick summary comprises the first few sentences of the result document.

Off

No summary is returned. This is the default setting for the **Summary** parameter.

Sentences=<length_of_summary>

If you have set **Summary** to **Concept**, **Context** or **Quick**, you can specify the number of sentences that you want the summary to comprise with the **Sentences** parameter. By default this is set to **5**.

For example:

http://1.23.45.67:2000/action=query&Text=perennial flowering plants&Summary=Concept&Sentences=3

DRE 4 configuration file settings

The following DRE configuration file settings allow you to manipulate the summaries that are produced as a result of queries to the DRE:

ContextSummaryQueryTermWeight

If you have included the **Method** parameter with the setting **Context** in a query/command, the DRE determines the weights to use for the terms in the query from the **ContextSummaryQueryTermWeight** setting in the configuration file. Enter a number for the weights that you want context queries to use. By default this is **255**.

MinWordsPerSentence

If you have included the **Method** parameter with the setting **Concept** in a query/command, the DRE uses the **MinWordsPerSentence** setting in the configuration file to determine the minimum number of words that a sentence must comprise in order to be considered as a sentence that can be used in the concept summary. The default value for this setting is **8**.

MaxWordsPerSentence

If you have included the **Method** parameter with the setting **Concept** in a query/command, the DRE uses the **MaxWordsPerSentence** setting in the configuration file to determine the maximum number of words that a sentence must comprise in order to be considered as a sentence that can be used in the concept summary. The default value for this setting is **35**.

Generating summaries during querying with DRE 3

You can include the following xoptions in the query string that you send to DRE 3 in order to generate summaries during querying:

http://<IP_address>:<query_port>/qmethod=q&querytext=<query string>&xoptions=<x_option>

xoptions=<x_option>

You can use the following values for **<x_option>** in order to specify the kind of summary that you want your query to return (the summary is returned in the **Summary** field:

withsummary

Returns a quick summary of the result document. A quick summary comprises the first few sentences of the result document.

For example:

http://1.23.45.67:2002/qmethod=q&querytext=perennial flowering plants&xoptions=withsummary

conceptsummaryN,LL

Returns a conceptual summary for each result document. The summary comprises sentences that are typical of the result's content (these sentences can be from different parts of the result document). The summary comprises **N** sentences and up to **LL** characters and is biased towards the key concepts in a document.

For example:

http://1.23.45.67:2002/qmethod=q&querytext=perennial flowering plants&xoptions=conceptsummary3,150

contextsummaryN,LL

Returns a conceptual summary for each result document that is biased by the context in the query text. The summary comprises sentences that are particularly relevant to the terms in the query (these sentences can be from different parts of the result document). It comprises **N** sentences and up to **LL** characters and is biased towards the context in the query text.

For example:

http://1.23.45.67:2002/qmethod=q&querytext=perennial flowering plants&xoptions=contextsummary3,150

6. Improving document content

Content breaking

It is usually advisable to break large documents into smaller chunks before they are indexed into the DRE. If you don't do this, it is more difficult to identify the relevant sections of documents - for example, if a 100-page document is returned as a query result, the user can't easily assess its relevance or find which section is relevant to their query.

Settings in the DRE configuration file and Import module settings in your Connector's configuration file allow you to break documents up before they are indexed.

Content breaking with the Import module

The following Import module settings in your Connector's configuration file allow you to control the way in which large documents are broken up during importing. If you break documents up, the smaller documents are indexed separately, but the DRE records that they are parts of the larger document.

ImportBreaking

Enter **true** if you want to split large documents into smaller documents before they are indexed (recommended).

If you enter **false** large documents are not split up into smaller documents, with the exception of the following (this is the default):

- HTML documents are split up at any anchors in the page, regardless of the number of words between the anchors.
- PDF documents are split up on every page regardless of the number of words on each page.

Note: if you enable **ImportBreaking** and you are using an earlier DRE version than DRE 4, you should set **Combine** to **1** in the configuration file of the DRE into which you are indexing the documents.

ImportBreakingMaxParagraphWords

If you have enabled **ImportBreaking**, **ImportBreakingMaxParagraphWords** allows you to specify the maximum size of the sections that a document is broken into. If a document is larger than **ImportBreakingMaxParagraphWords**, the Import module splits the document at the first sentence end after **ImportBreakingMinParagraphWords**, or at **ImportBreakingMaxParagraphWords** if no sentence end is found earlier. By default, the value of this setting is **360**.

ImportBreakingMinParagraphWords

If you have enabled **ImportBreaking**, **ImportBreakingMinParagraphWords** allows you to specify where you want section breaks to occur. The Connector splits documents at the end of the first sentence after the number of words you specify with **ImportBreakingMinParagraphWords**, or at **ImportBreakingMaxParagraphWords** if no sentence end is found earlier. By default the value of this setting is **160**.

ImportBreakingMinDocWords

Enter a number of words that a document must contain to be split up into smaller documents (provided **ImportBreaking** is enabled). Any document that contains less than the number of words you specify is imported whole. The default value for this setting is **360**.

Note:

- The Connector attempts to split up the document at the end of a sentence within the range you specify with **ImportBreakingMinParagraphWords** and **ImportBreakingMaxParagraphWords**.
- **ImportBreakingMinDocWords** must be greater than **ImportBreakingMinParagraphWords**.

Example configuration

The following is an example of settings in a Connector configuration file, which sets up content breaking during importing:

ImportBreaking=true

ImportBreakingMinDocWords=500

ImportBreakingMinParagraphWords=400

ImportBreakingMaxParagraphWords=600

In this example, **ImportBreaking** is enabled, so the Import module breaks all documents that contain more than **600** words into sections. A new section starts at the end of the first sentence after **400** words, or at exactly **600** words if there is no sentence end earlier.

Content breaking with the DRE

If your DRE version is DRE 4, the following settings allow you to control the way in which large documents are broken up during indexing.

MaxFieldLength

If a field contains less than the specified number of characters it is not broken into sections. The default value for this setting is **100**.

MaxSectionLength

Specify the maximum number of characters that a document can contain before it is broken into sections. The default value for this setting is **1000**.

Good content

The configuration of the Connector to retrieve good, contextually rich content is vital. Connectors are highly configurable to allow you to ensure good content is aggregated, and irrelevant or misleading information (for example, tables of contents) is ignored. Settings in the Import module and DRE configuration files allow you to specify the following:

- Minimum and maximum size that documents must be in order to be indexed
- Strings, HTML tags or number of words that indicate the start and end of document content that you want to index
- Tags or strings that delimit the start and end of sections of document content that you want to index
- Strings that a document must contain in order to be indexed
- Strings that a document must not contain in order to be indexed
- White space or specified strings that you want to remove from documents before they are indexed
- XML tags that you want to remove from documents before they are indexed
- Strings or tags that you want to replace with other strings
- Date formats that should be recognized during indexing

Please refer to the DRE and Import module documentation in order to determine which settings you can make for the documents you want to index.

Proper names and extended concepts

If your DRE version is DRE 3, you can set the DRE to index names that consist of several words as one word with the following parameter in the DRE configuration file:

ProperNames (DRE 3)

Enter **on** to index names that consist of more than several words as one word. The name **William Shakespeare**, for example, would be indexed as one word. This is the default setting.

Enter **off** if you want all names to be indexed as individual words.

If your DRE version is DRE 4, this functionality is also applied to other extended concepts:

ProperNames (DRE 4)

Enter one of the following to determine how the DRE treats terms that form a unit (for example, "William Shakespeare" or "heart attack":

0 The DRE stores each term individually. (For example, it stores "William", "Shakespeare", "heart", and "attack".)

1 The DRE stores each term individually and as a unit, if the terms in the unit begin with capital letters. (For example, it stores "William", "Shakespeare", "William Shakespeare", "heart" and "attack".)

2 The DRE stores each term individually and as a unit. (For example, it stores "William", "Shakespeare", "William Shakespeare", "heart", "attack" and "heart attack".)

Stop lists

Stop lists, which are lists of words that should be ignored, allow you to specify words and phrases that should be removed from the content that is indexed because they lack meaning. You do not normally need to edit the stop list that is provided with your Autonomy product, but it can be helpful to add unwanted words and phrases that occur regularly in your documents to the stop list.

This is particularly beneficial for the stop lists that Categorizer or Classification Server uses. For example, if all your documents include a copyright statement, you could add this to the stop list to which Categorizer or Classification Server refers in order to prevent it from being included in taxonomies that the products generate.

Languages

If you want to implement stemming with DRE 3, you must ensure that you set the following parameters correctly, so that the stemming algorithm and DRE language match:

StripLanguage	Implements stemming and specify the stemming algorithm
CharConv	Sets the language in which the DRE is running

Please refer to your DRE documentation for a list of the values that these parameters should take.

Note: if your DRE version is DRE 4, you do not need to make these settings. Where stemming is available, the correct algorithm is automatically selected when you specify the DRE language and set **Stemming** to **true** in the **[LanguageTypes]** section of the DRE configuration file.

Indexing metadata

You can configure the Connector you are using to retrieve data from your repositories to extract metadata, which you can store in fields alongside documents that you index into the DRE. It is sometimes tempting to store all metadata for a document just in case it is needed. This is inefficient: it increases document sizes, which slows down importing and indexing, and unnecessarily increases the size of the DRE, which slows down querying. Instead, you should consider the kinds of query that will be sent to the DRE, and the metadata that you need to display, and index only the metadata that is needed.

In addition to indexing only the metadata that you need, it is important for query performance that you index metadata into the most useful kind of field. The following make querying more efficient:

- index fields
- numeric fields and numeric date fields

Setting up index fields

If a field is going to be used frequently, you can set it up as a DRE index field (using the DRE's configuration file) in order to optimize the query process when you restrict queries using these fields. This means that the field is stored in the DRE's internal files, which enables faster access to the data in the field. 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.

Please refer to the **Storing and indexing fields** section in the **Configuring the DRE** chapter of the DRE manual for full details of how to add a field process to the DRE configuration file in order to set up index fields.

Setting up numeric fields and numeric date fields with memory mapping

You can configure the DRE to identify fields that contain numerical values. General numeric values are stored in numeric fields, and date values are stored (as epoch seconds) in numeric date fields. When these fields are queried, the DRE stores them in a fast look-up table in memory (in the **numeric** subdirectory of the main DRE installation directory) so it can quickly return the field.

Fields that the DRE identifies as numeric fields or numeric date fields are used during the pre-filtering stage of the query process, to reduce the number of documents that the DRE searches for conceptual matching. (Please refer to the **Optimizing querying** section of the **Improving DRE performance** chapter for details of the query process.)

Please refer to the **Setting up memory mapping to speed up numerical queries** section in the **Configuring the DRE** chapter of the DRE manual for full details of how to implement memory mapping for your fields.

7. Configuring data storage

Checking your data is indexed into the DRE

If you are encountering problems with indexing data into the DRE, you can perform the following actions in order to check on the progress of the importing and indexing processes.

1. **Check the Connector's log file**

   A Connector's log file tells you how many files are being passed to the importing stage and gives a reason for certain files are being ignored.

   If some of your files are not reaching the importing stage, ensure that the following are set up correctly in the Connector's configuration file:

   - the security details that you set for the Connector must allow the Connector to access all the documents that you want to index
   - documents must match the configuration file's settings for document size and format

2. **Create an Import log file**

   You can view a log of the importing process when you run a Connector, which tells you which files are failing to import and for what reason. In order to create an **import.log** file which shows the details you need to see, make the following settings in the Connector's configuration file:

   ImportLogFile=import.log

   ImportLogLevel=full

   ImportLogFileAppend=true

   Please refer to the Import module documentation for details of these settings.

3. **Create a structured IDX file without indexing documents into the DRE**

   You can check whether all your files are making it into IDX format by creating a structured IDX file without indexing documents into the DRE. This allows you to view all the documents which have successfully passed through the importing process.

   Run the Connector once on a sample of your data. Ensure that the following settings are made in the Connector's configuration file:

   - use false settings for the DRE into which the Connector indexes data
   - set the Connector to cycle just once

- set **BatchProcess** to **Import**
- set **MapAll** to **true** (this means that document fields are mapped to DRE fields of the same name, which enables you to see the original document field names in the IDX file)

When you run the Connector, an IDX file (or TMP file if you are using AutoIndexer) is written to your working directory, or to your index directory if you have specified one in the configuration file.

For example, if you are using Notes Fetch, you should make the following settings in the **[Default]** section of the configuration file:

DRE=1.1.1.1

SpiderCycles=1

BatchProcess=Import

MapAll=true

4. Check the DRE log file

If your documents are being imported successfully, the DRE log file should contain details of problems encountered during the indexing process, for example your documents may be being overwritten or rejected by some DRE configuration setting.

You should also check that your documents are not being expired too quickly. Please to refer to the **Setting expiry time** section in this chapter for details of how to set up expiry correctly.

Removing duplicates during indexing

To make the most efficient use of your system, you can remove duplicate documents during the indexing process. This maintains the quality of the data in your DRE and improves the efficiency of queries, but may slow down the indexing process.

By default, any duplicate documents are indexed into the DRE, but you can set the DRE to look for duplicates at index time, remove any that it finds and index only the new document with the **KillDuplicates** parameter. The **KillDuplicates** parameter allows you to specify which fields must match in order for two documents to be considered duplicates. The way in which you can set this depends on what DRE version you are using.

Removing duplicates with DRE 4 / AXE

In DRE 4 / AXE, you can use the **KillDuplicates** parameter both in the configuration file and as an optional parameter for the **DREADD** command.

For example in the configuration file:

KillDuplicates=<value>

For example with the DREADD command:

http://<IP_address>:<index_port>/DREADD?<file_name>&KillDuplicates=<value>

Note that this setting overrides the setting in the configuration file for the documents you specify with **DREADD**.

<IP_address>

Enter the IP address of the DRE into which you want to index your data.

<index_port>

Enter the port number for the DRE into which you want to index your data.

<file_name>

Enter the name of the file that you want to index.

<value>

Enter one of the following values for **KillDuplicates**:

NONE

The file is indexed and the DRE does not check for duplicates. This is the default.

REFERENCE

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

REFERENCEMATCH*nn*

If a document is indexed whose content is more that **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.

Note: this setting slows down the performance of your DRE, and should be avoided unless Autonomy has advised you to use it. If you do use this setting, the **nn** value should normally be greater than **90 %**.

<FieldName>

If a document is indexed that contains a **<FieldName>** field with 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.

For example:

DRE 4 example:

KillDuplicates=*/DOCUMENTID

In this example, if the value of the **DOCUMENTID** field for a new document matches the value of the field for an existing DRE document, then the DRE deletes the document that it already contains.

Removing duplicates with DRE 3

In DRE 3, **KillDuplicates** is an optional parameter for the **DREADD** command.

For example:

http://<IP_address>:<index_port>/DREADD?<file_name>&KillDuplicates=<value>

<IP_address>

Enter the IP address of the DRE into which you want to index your data.

<index_port>

Enter the port number for the DRE into which you want to index your data.

<file_name>

Enter the name of the file that you want to index.

<value>

Enter one of the following values for **KillDuplicates**:

NONE

The file is indexed and the DRE does not check for duplicates. This is the default.

REFERENCE

If data is indexed that has the same **DREREFERENCE** as a document that the DRE database into which you are indexing already contains, the DRE deletes the document that it already contains and replaces it with the new one. If you want to check all the databases in the DRE, enter **REFERENCE2**.

REFERENCEMATCHnn

If a document is indexed that has the same **DREREFERENCE** as a document that the DRE database into which you are indexing already contains or if its 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. If you want to check all the databases in the DRE, enter **REFERENCE2MATCH*nn***.

Note: using **ReferenceMatch*nn*** will slow down the indexing process as the DRE has to check the content of all documents that have the same DREREFERENCE as the one that you are indexing.

MATCHnn

If a document is indexed whose content 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.

Note: using **Match*nn*** will slow down the indexing process as the DRE has to check the content of all the documents it contains.

<FieldNameA><FieldNameB>…

Allows you to specify one or more fields that the document that you are indexing contains. If the DRE already contains a document that has the same fields, the DRE deletes the document that it already contains and replaces it with the new one.

For example:

KillDuplicates=_DOCUMENTID_DRETITLE

In this example, if the value of the **DOCUMENTID** and **DRETITLE** fields for a new document match the value of the fields for an existing DRE document, then the DRE deletes the document that it already contains.

If you want to check all the databases in the DRE, you should use the following format:

KillDuplicates=_<FieldNameA>2_<FieldNameB>

For example:

KillDuplicates=_DOCUMENTID2_DRETITLE

Note:

- If you want to specify multiple fields, you must separate them with underscores (there must be no space before or after an underscore). You must also enter an underscore before the first field that you are specifying.
- Using the **_<FieldNameA>_<FieldNameB>…** parameter will slow down the indexing process as the DRE has to check the fields of all the documents it contains.
- Any fields that you use here must be declared as reference fields in the DRE configuration file.

Setting expiry time

If you have not set your Connectors to update the data in the DRE, you must set the DRE configuration file to not expire data or you will lose your data. Set **Expire** in the **[Schedule]** section of the DRE configuration file to **false**.

If you set **Expire** to **true**, you can set **ExpireTime** in the **[Databases]** section of the configuration file with the number of hours before data expires, or (if your DRE version is DRE 4) with **0** if you don't want data to expire from that database.

For DRE 4, you can specify expiry time per document by setting **ExpireDateType** in the **[Properties]** section of the configuration file to **true** (use **PropertyFieldCSVs** to specify the fields that hold the expiry dates for your documents).

8. Setting up security

For details of how to set up security in Autonomy products, please refer to the **IAS manual**. (This includes a description of the differences between security in DRE 3 and DRE 4.)

This chapter outlines some general security considerations for setting up your Autonomy software infrastructure.

Performance

- Restrictive security in documents can slow down DRE query speed, even if you have implemented mapped security. You can significantly reduce the loss of performance of your system by distributing documents between multiple DREs.

- Connector operations with NT and NetWare can be slowed down by the ACL extraction for each document if you have implemented mapped security for these document types. You should bear this in mind when determining how frequently you want to update DRE documents.

- Queries can be slowed down by incorrect security settings for a user. In order to reduce time wastage here, ensure that user security roles and groups are set up correctly in the front-end application (if you are using Portal-in-a-Box) and UAServer. For example, a user who has no access to a particular file store should be excluded from the store by their portal role.

- If your DRE version is DRE 3, ensure that the ACL field is memory mapped in order to speed up access to ACLs. You should also make sure that you have set the size of the ACL field to be large enough for the maximum possible ACL size. You can do this by setting the following parameters in the DRE configuration file:

  [Server] section

  MemFields

  Set to **true** to allow the fields you specify in the [Fields] section of the configuration file to be stored in memory for increased query speed.

  [Fields] section

  nn=MEM.size,name

  Replace **name** with the name of a field that you want to hold in memory. Only the first **size** bytes will be stored in RAM.

Integrating Autonomy products

- If your front-end application is Portal-in-a-Box, and you are using multiple DREs with a DQH, ensure that the database mappings between the components are identical so that the security strings that are constructed by the Portal can map back to the DRE databases successfully. In order to achieve this, it is advisable that you have identical database names for each content store. This does not apply if you have set up security by database (rather than by document; see the IAS manual for details).

- If you have multiple secure sources in your installation, it is likely that user details required for security will differ between sources. For example, user names for NT are not be the same as for Lotus Notes or Exchange, and so on. If your front-end application is Portal-in-a-Box, you should select one user name as the base by which the Portal identifies the user (NT user name is simplest because Portal-in-a-Box can create this user automatically). The user can then enter other user names manually, when they are required; or you can set a list of user details for the various sources with the following parameter in the UAServer configuration file:

  SecurityRepository*n*=<security_repository>

  If you use UAServer to store a user's security details for a repository, **SecurityRepository*n*** allows you to edit the value that is stored as the repository name. The string that you enter for this must be the name of the security section that contains the security settings for this repository in the DRE configuration file. You can use **SecurityUserName*n***, **SecurityPassword*n***, **SecurityDomain*n*** and **SecurityGroup*n*** to edit the user's security details for the repository (which of these details are required depends on the repository).

  If you want to edit multiple sets of security details, you must number them. The numbering must start from **0** and be consecutive.

  For example:

  SecurityRepository0=NT&SecurityUserName0=JohnSmith&SecurityPassword0=mypassword&SecurityGroup0=marketing,sales&SecurityRepository1=Exchange&SecurityUserName1=JohnS&SecurityDomain1=MyCompany

- If you are using Portal-in-a-Box with ActiveKnowledge, you must ensure that the following actions are taken when you set up the installation:

  If you are using ActiveKnowledge to communicate with a secure web server with NT authentication, set **UseNTLM=true** in the ActiveServer configuration file.

  The user must log into the Portal in order to set the security permissions in the user's security details file.

9. Querying a DRE

There are a number of types of query that you can send to the DRE, and for each of them you can use field specifiers to restrict the results that are returned and displayed. Which options are available and how you should implement them in your queries depend on what version of the DRE you are using. Please refer to your DRE documentation for further details.

Query types

Once you have configured the DRE, 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. (Note that you need to enable fuzzy queries before you can use them.)

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 configurable Soundex algorithm).

Proper names

Names are recognized and treated as a unit.

Restricting queries with field specifiers

For all of the query types, you can restrict the result documents that the DRE returns by specifying values that fields in the result documents must have. You can also send field queries to the DRE independently in order to return all documents that match the field criteria you specify.

The field specifiers you can use include the following:

String match	Field values must exactly match the string you specify (you can also perform string matches that include a wildcard character).
Term match	Field values must match the concepts in the string you specify.
Fuzzy match	Field values are fuzzy matched to the string you specify.
Alphabetical range	Field values must lie within the range you specify.
Date range	Field values must lie within the range you specify.
Numerical range	Field values must lie within the range you specify.
Comparative numerical	Field values are greater than / less than / equal to / not equal to the value you specify.
Comparative date	Field values are before or after a time you specify.

The DRE performs the following steps for queries containing field specifiers:

- The DRE constructs a list of possible results for the input.
- For each of the possible result documents, the DRE determines whether its fields match the restrictions you have specified. If a document's field matches the value specified, the document is added to the final results list.

The first stage is extremely fast, as the DRE executes a standard query. The second stage is sequential and its duration depends on the number of possible results that the first stage identified. In order to achieve optimum performance for queries with field restrictions, you must construct the query to return a small set of possible results in the first stage. For example, the following two queries will return the same results, but the second can be completed more quickly:

DRE 4:

Action=Query&Text=drama&FieldText=Shakespeare:*/Author

Action=Query&Text=drama+AND+(Shakespeare:Author)&FieldText=Shakespeare:*/Author

DRE 3:

qmethod=q&querytext=drama&fnameAuthor=Shakespeare

qmethod=q&querytext=drama+AND+(Shakespeare:Author)

In the first query, result documents that contain concepts that match **drama** are returned as possible results. The DRE checks the ***/Author** field of each of these in order to find fields that contain exactly the value **Shakespeare**, and returns these as query results.

In the second query, result documents that contain **drama** and where the **Author** field contains **Shakespeare** are returned as possible results. This will usually generate a shorter list of possible results, so than the first query so the DRE can more quickly check through the ***/Author** field of the possible documents in order to find only documents where the ***/Author** field contains exactly the value **Shakespeare**.

Note: If you want to send a query with the field restriction **(Shakespeare:Author)**, you must declare the **Author** field as an Index field in your DRE configuration file.

For full details of how to use field specifiers and field restrictions in order to optimize your queries, please refer to your DRE documentation.

Using field restrictions to display results

When it returns query results, the DRE by default returns the fields that you have set up as Reference fields in the DRE's configuration file.

If your DRE version is DRE 4, you can alter which fields are displayed for result documents, by adding the following parameters to your action command:

Print	Allows you to return all explicitly indexed fields, the content of the index fields as a single text field or all XML fields.
PrintFields	Allows you to specify fields that you want to display. **Note:** you can only specify fields that are declared as index fields in the DRE configuration file.

10. Improving DRE performance

There are several factors that you can change in order to enable the DRE to make the best use of your system resources:

- running indexing and query processes at different times
- number of CPUs per DRE
- number of threads that the DRE uses
- amount of RAM allocated to index and term caches
- quality of DRE content
- query type
- delayed synchronization during indexing

In addition, you can improve performance by using multiple DREs with the Distributed Index Handler (DIH) and Distributed Action Handler (DAH). Please refer to the **Planning, installing and configuring Portal-in-a-Box 4** chapter for details of how to determine the best distributed architecture for your needs.

Scheduling indexing and query processes

The main tasks the DRE performs are:

Indexing

Documents in IDX or XML file format are added to the DRE. You can schedule when indexing is performed according to your needs. This depends on the kind of system you are setting up: for some systems, new data must be added to the DRE as soon as it becomes available; for other systems, weekly updates may be enough.

Querying

Users send natural language, keyword or Boolean queries to the DRE, which analyzes the concept of the query and returns documents that are conceptually similar. The user requires a real-time response to the query, so query speed is particularly important.

Indexing and querying both use the DRE's resources heavily, so it is best to run them at different times. For example, if you are setting up a DRE that users will send queries to during office hours, you should schedule indexing tasks to run during the night. However, in some circumstances, this is not possible, and the recommendations in this chapter include suggestions of how you can balance indexing and querying tasks.

Calculating the number of CPUs needed per DRE

The DRE should have exclusive access to at least one CPU. Increasing the number of CPUs that a DRE uses increases the speed of DRE operations. If you are setting the DRE up as part of a system in which indexing is carried out at the same time as users are sending queries to the DRE, you should make at least 2 CPUs available to the DRE, so that one CPU can always be dedicated to indexing. This is because indexing is resource-intensive, and with just one CPU for both indexing and querying, query speed may become unacceptably slow.

Note that ideally, the number of CPUs that the DRE has access to should always be high enough to enable you to set the number of threads available for query commands to at least 4. (Please refer to the **Calculating the number of threads that the DRE can use** section for details of how to calculate how many query threads to set.)

Calculating the number of threads that the DRE can use

The DRE is a multi-threaded process: it can execute multiple commands at the same time, which enables it to maximize the utilization of the CPU and disk resources. For example, during the lifetime of a query, time is spent on disk (reading information needed to resolve the query) and in the CPU (processing values and relationships in order to determine which information to return). With multi-threading, rather than completing all stages of one query before moving onto the next, the DRE can complete parts of one query that use the CPU, whilst retrieving information from disk for another query, or writing information to disk for an indexing command. This enables more efficient use of resources, since there is less time when the CPU and disk are not being used.

DRE threads

The DRE uses different threads for indexing, querying and various other processes (such as executing service commands and gathering statistics). Of these threads, indexing and querying make the heaviest use of system resources. Indexing is single threaded, but the DRE can use multiple threads for querying, and you can configure the number of query threads that the DRE can use, according to how your system is going to be used.

Calculating the optimum number of query threads for your DRE

The balance of CPU and disk usage for the DRE means that use of resources is maximized during querying when a CPU deals with between 1 and 2 threads. For example, if your DRE has access to 4 CPUs, it should use between 4 and 8 query threads. If too few threads are used, resources will be underused. If too many threads are used, the CPU resources needed for context-switching (when the CPU changes its attention from one thread to another) mean that no benefit is gained from the high number of threads.

An additional consideration when calculating the number of query threads the DRE can use is the kind of query that is going to be sent to the DRE. If your users will be sending queries such as keyword queries, which can be processed very quickly, you should set the number of threads with a low value (because for fast queries, there is less time when the CPU and disk are idle). If your users

will be sending slower queries, such as queries with multiple field restrictions that take more than one second to resolve, you should set the number of threads to a high value (because for slower queries, there is more time when the CPU and disk are idle). Please refer to the **Optimizing querying** section for details of which queries you can expect to be executed quickly or slowly.

Note: ideally, your DRE should have access to enough CPUs to allow you to set the number of threads available for query commands to at least 4.

Balancing threads between indexing and querying tasks

Where possible, you should set up indexing tasks to be performed at a time when queries are not being sent to the DRE, or when query volume is low. This means that the maximum resources can be available for handling queries, where speed is most crucial.

If indexing and query commands are being sent at the same time, your calculation of how many threads to set for querying should take this into account. You should allow one CPU to be used for indexing only, and make your calculation of how many query threads the DRE can use accordingly. For example, if your architecture has 4 CPUs for a single DRE, you should assume that one CPU will be used for indexing, and calculate the number of query threads to use as if there were just 3 CPUs. That is, query threads should be set between 3 and 6.

To set the number of threads that the DRE uses for queries:

Parameters in the [Server] section of the DRE configuration file allow you to specify maximum number of query threads that the DRE uses. The way you set these parameters depends on whether you are sending **action** commands (for example, action=Query&Text=accounts) or **qmethod** commands (for example, qmethod=q&querytext=accounts) to the DRE.

If you are sending **action** commands to the DRE, the following parameter allows you to specify the maximum number of query threads to use:

MaximumThreads

The number of ACI action commands that the DRE can process in parallel at any one time. The default value for this setting is **10**.

Note: if you are sending action commands to the DRE, the **QueryThreads** parameter is still set, but you can exclude the threads that it specifies from your calculation of the number of query threads that your DRE is using, since the threads that **QueryThreads** specifies will never be used.

If you are sending **qmethod** commands to the DRE, the following parameter allows you to specify the maximum number of query threads to use:

QueryThreads

The number of qmethod queries that can be processed in parallel at any one time. The default value for this setting is **4**.

Note: if you are sending qmethod commands to the DRE, the **MaximumThreads** parameter is still set, but you can exclude the threads that it specifies from your calculation of the number of query threads that your DRE is using, since the threads that **MaximumThreads** specifies will never be used.

File descriptor limitations on Solaris

The Solaris operating system allows a maximum of 256 file descriptors per process. This is lower than other operating systems (generally around 1024), and puts limits on DRE size and number of threads.

File descriptor usage for DRE version 4 is as follows:

- 25 static file descriptors (for socket handling, log files and other administrative processes)
- at least 6 file descriptors for each thread (for dynterm and nodetable files)

For a small DRE, there are just 2 dynterm files and 4 nodetable files, but as the DRE increases in size the number of dynterm and nodetable files increases. (When the size of the dynterm or nodetable file reaches the internally set size limit for these files, the DRE starts a new file. This size limit is generally 2 gigabytes for Solaris, and 1 gigabyte for other operating systems.) For each additional dynterm or nodetable, each thread needs to use an additional file descriptor.

This means that there is a limit on the DRE size and number of threads that can be used - as each extra thread uses multiple extra file descriptors. On most operating systems, this limit is rarely reached (the number of file descriptors needed rarely exceeds about 500, which is within the 1024 maximum for most operating systems). On Solaris, the 2 gigabyte limit on dynterm and nodetable file size means that the DRE needs fewer file descriptors per thread than on other operating systems (which have a 1 gigabyte limit), but the low limit on file descriptors should still be considered when you are calculating the size of DREs and number of threads to use.

For example, a DRE that uses up to 8 threads, can have a maximum size of 30 gigabytes (around 7 million documents).

Calculating cache sizes

Indexing and querying make use of cached data to speed up the processes. DRE configuration parameters allow you to set the maximum size that these caches can reach. When you calculate this size, you should ensure there is sufficient RAM available to the DRE for the caches. If there is not enough RAM available, the data storage may be switched from RAM to disk, which causes a significant performance reduction.

The balance between the size of the caches for indexing and querying depends on how the DRE is being used. Both caches should usually be set to be as large as possible, but the total size of the caches should never exceed 65% of the total available RAM.

Index cache

For indexing, the index cache is used to hold a representation of the data that is going to be indexed into the DRE. It is particularly important for DRE performance that this cache is large enough to hold the data that you are indexing. The amount of data that the index cache needs to hold depends on how much data you are indexing into the DRE at a time.

In order to calculate the amount of data that your index cache needs to hold, you need to consider how you have set the **DelayedSync** parameter (please refer to the **Optimizing indexing** section for details of how to use **DelayedSync**):

- If you are setting **DelayedSync** to **false**, the index cache only needs to be large enough to hold one file, because each file is synchronized with the DRE (and the cache is flushed) as soon as the data for a file has been added to the index cache. In this case, you should set the size of the index cache to be slightly larger than the largest file you expect to index into the DRE.
- If you are setting **DelayedSync** to **true**, the index cache needs to be large enough to hold all the files that will be added to it before the timeout period you specify with the **MaxSyncDelay** parameter, because the files are only synchronized with the DRE (and the cache flushed) after the timeout period has been reached. Note that files in the index cache will also be synchronized with the DRE if it gets full before the timeout period (which will affect your calculations of the way system resources are used).

The following parameter in the [IndexCache] section of the DRE configuration file allows you to specify how much memory the DRE can use to cache data for indexing:

IndexCache

The size of the index cache in kilobytes. The default value for this setting is **102400**.

Term cache

For querying, the term cache stores recently used query terms. A larger term cache can increase the speed of query resolution.

The following parameter in the [TermCache] section of the DRE configuration file allows you to specify how much memory the DRE uses to cache query terms:

TermCache

The size of the term cache in kilobytes. The default value for this setting is **102400**.

DRE content

The way you configure the DRE content can affect DRE performance, particularly during querying. The following can enable the DRE to run more efficiently:

- avoid duplication
- select index fields carefully
- return only the query fields that you need to display

Avoiding duplication

Duplicate data in the DRE can slow down query performance. The DRE is optimized internally for situations in which the content is statistically representative of the corpora from which queries will be performed. If data is duplicated, the distribution of terms is misleading (terms that are statistically rare can end up being indexed an unrepresentatively large number of times). This means that the occurrence of query terms can appear more frequent than it actually is, which slows down querying.

In addition, extra occurrences of query terms can cause the term cache to fill up quickly, which results in decreased performance since fewer useful query terms can be cached.

Please refer to the **Removing duplicates during indexing** section of the **Configuring data storage** chapter for details of how to remove duplicates during indexing.

Selecting fields to index

You can configure the Connector you are using to retrieve data from your repositories to extract metadata, which you can store in fields alongside documents that you index into the DRE. It is can be tempting to store all metadata for a document just in case it is needed. This is inefficient: it increases document sizes, which slows down importing and indexing, and increases the size of the DRE, which slows down querying. Instead, you should consider the kinds of query that users will send to the DRE, and the metadata that you need to display, and index only the metadata that is needed.

It is also important for query performance that you index metadata into the most useful kind of field. The following can be useful in making querying efficient:

- index fields
- numerical date fields

Please refer to the **Indexing metadata** section of the **Improving document content** chapter for more details.

Selecting fields to display

You should also ensure that queries only return the metadata that you need, rather than returning all fields and leaving it up to the front-end application to select which metadata is displayed. See the **Using field restrictions to display results** section of the **Querying a DRE** chapter for details of how to specify the fields that the DRE returns for query results.

Optimizing querying

The query process

The query process works in 3 stages:

1. Pre-filter

   At this stage, database restrictions, date-field restrictions, and numerical field restrictions that you specify in the query string and configuration file are processed. This reduces the amount of DRE content that the query runs against, which speeds up the query process.

2. Conceptual matching

   The DRE finds conceptual matches for the query within its content.

3. Post-filter

   At this stage, the DRE checks the results that it found during conceptual matching to ensure that they match any field text restrictions that you included in the query string. This is relatively time-consuming, and multiple field restrictions at this stage can slow down the query process.

The recommendations that this chapter makes for allocating query threads and scheduling indexing tasks for times when users are not sending queries to the DRE can improve the speed of the query process. In addition, when you are optimizing query speed, you should consider the kinds of query that you are sending to the DRE.

Query types

Standard queries, such as keyword and conceptual searches are significantly faster than queries that use field restrictions, especially when field restrictions apply to text fields that have not been set up as Index fields. If you are querying with field restrictions, you can reduce this effect by ensuring that the fields that you are querying on are correctly indexed (please refer to the **DRE content** section of this chapter for further details, and to the **Querying a DRE** chapter for an outline of query types).

Optimizing indexing

The speed of the indexing process is usually less critical than the speed of the query process, but with large amounts of data being indexed into the DRE, it is still important to improve the efficiency of the process where possible. In addition, the way you configure the indexing process can have effects on the efficiency of the query process.

The indexing process

The indexing process works in 2 stages:

1. The DRE creates a representation of the new data in the index cache.
2. The cache is synchronized with data that the DRE currently contains, and the new data is stored on disk and removed from the index cache.

When you are scheduling indexing, you should consider this chapter's recommendations on DRE content (particularly on selecting fields to be indexed), and on running indexing and querying processes at different times. In addition, the delayed synchronization feature allows you to change the stage at which the index cache is synchronized with the DRE, depending on whether your priority is achieving fast query speeds or making new information available to the user as quickly as possible.

Delayed synchronization

The delayed synchronization feature allows you to select how the index cache is synchronized with the DRE data. This is useful in systems where indexing tasks are scheduled at times when the DRE is also handling queries.

By default, synchronization happens as soon as a representation of data has been made in the index cache. New data is available to the user (as query results) quickly, so you should use this setting in systems where up-to-date data is the priority. But synchronization uses resources that the DRE could otherwise be using for querying. Delayed synchronization reduces the impact of this effect by collecting multiple data representations in the index cache and then synchronizing them all with the DRE data in one go. This is useful in systems where query speed is more important than having up-to-date data.

Note: delayed synchronization is recommended if you are indexing a lot of small files (files that are smaller than 100MB).

The following parameter in the [Server] section of the DRE configuration file allows you to specify whether the indexing process uses delayed synchronization:

DelayedSync

Enter **true** if you want the DRE to delay synchronization. If you set **DelayedSync** to **true**, the DRE only stores data on disk when:

- the index cache is full
- the index cache contains some data and the timeout specified by **MaxSyncDelay** has expired

11. Clustering with Classification Server

Classification Server allows you to take a snapshot of the data in a DRE. This snapshot identifies clusters of conceptually similar documents, which enables you to generate a view of trends in the data. You don't need to generate an initial taxonomy in order to take a snapshot.

A set of data can contain a few large clusters or many small clusters, as well as a number of outliers that aren't part of any cluster. Clusters may consist of highly similar documents or of less closely related ones. What constitutes optimal clustering depends to some extent on how you intend to use your clusters, but the aim of clustering is always to generate an accurate characterization of the data in your DRE.

By default Classification Server uses internal settings to produce clusters. These default settings do not usually need to be changed, but in some cases you may require more or less detail in your clusters, or the amount and nature of your data may mean that default clustering is not satisfactory. You can optimize clustering in these cases by setting parameters that adjust the size of the units on which clusters are based, the degree of conceptual similarity that documents within clusters must have, or the number of clusters that are created.

The clustering process

There are two main stages to the clustering process:

- building "seeds"
- grouping seeds into clusters

Seed-building is implemented when the **ClusterSnapshot** action is executed. Classification Server takes a sample of the documents in the Data DRE and tries to associate each of these documents with other documents in the DRE - based on the similarity of the concepts that the documents contain. Each of the groups of sample document and similar documents produced at this stage is a seed. Classification Server stops trying to build a seed when the seed meets the requirements that **SeedSize** specifies or when there are no more documents in the DRE that meet the similarity requirement that **SeedBindLevel** specifies (whichever condition is reached first). Classification Server discards any seeds that don't reach the required size. The number of clusters you specify with **NumClusters** affects the number of sample documents from which Classification Server tries to create seeds at this stage (note that you can adjust the relationship between the number you specify here and the size of the sample used by changing the value of **StartingSuggestOverrideFactor)**.

Grouping seeds into clusters is implemented when the **ClusterSGDataGen** or **ClusterCluster** actions are executed. Classification Server tries to create clusters by grouping seeds together. The grouping is based on the similarity of the concepts that the seeds or clusters contain. Clustering is complete when the number of clusters specified by **NumClusters** has been created, or when no more clusters can be created that meet the similarity requirement specified by **BindLevel** (whichever condition is reached first). Clusters that don't meet the quality requirement set by **BindLevel** or the size requirement set by **MinClusterDocs** are discarded.

Optimizing clustering

You can edit the following settings in the **[Cluster]** section of the Classification Server configuration file in order to optimize clustering (most of the settings can also be included in action commands as indicated in the description of the setting):

SeedSize

Specify the size of the seed around which clusters are formed in a Snapshot. A seed is the unit that a cluster is based on, and is built from documents in the Data DRE that contain similar concepts. **SeedSize** represents the size of the document group that is required in order for the documents' concepts to form the basis of a cluster (note that **SeedSize** does not specify an absolute value for the number of documents in a seed). You can use **SeedBindLevel** to specify the conceptual similarity that the documents must have to each other in order to form a seed. **-1** uses an internal default for the seed size.

You can set this parameter in the configuration file, or include it in a **ClusterSnapshot** action command.

SeedBindLevel

Specify how tightly bound to each other the concepts between the documents that form a seed must be (a seed is the unit around which a cluster is built). **-1** uses an internal default for the bind level (a higher value here gives more tightly bound seeds).

You can set this parameter in the configuration file, or include it in a **ClusterSnapshot** action command.

BindLevel

Specify how tightly bound the concepts within clusters are. **-1** uses an internal default for the bind level (a higher value here gives more tightly bound clusters).

This parameter applies when seeds are grouped into clusters; it specifies the similarity between seeds in order for them to be grouped into a cluster.

You can set this parameter in the configuration file, or include it in a **ClusterCluster** or **ClusterSGDataGen** action command.

StartingSuggestOverrideFactor

Enter a value in the range **1** to **10**, in order to specify how large a part of the DRE should be clustered (a higher value allows a larger part of the DRE to be clustered). The default value for this setting is **1**.

Note: it is strongly recommended that you don't set **StartingSuggestOverrideFactor** to a value greater than **3**, unless you are advised to do so by Autonomy.

NumClusters

The maximum number of clusters that you want to be generated. The default value for this setting is **25**.

This parameter determines how detailed your view of the data in the DRE is. Higher values for **NumClusters** give a more detailed, low-level view; lower values give a less detailed, high-level view.

You can set this parameter in the configuration file, or include it in a **ClusterCluster** or **ClusterSGDataGen** action command.

MinClusterDocs

The minimum number of documents that a cluster is allowed to have. Clusters that don't meet this requirement are discarded. The default value for this setting is **10**.

Configuration recommendations

The ideal values for the parameters that affect clustering depend on the nature and amount of data in your DRE. It is possible to make some general recommendations about how to change these parameters according to your data. Parameters are closely interdependent, so you should make these changes in combination with each other (rather than just changing one of the settings), and change values in small increments or decrements.

Note: although you can make many changes to clustering, the number and size of clusters that Classification Server can identify depends ultimately on the data that your Data DRE contains.

Small amount of data

If your DRE has a small amount of data, Classification Server is likely to identify fewer clusters, since it is less likely that your data will contain a lot of similar documents for a number of different topics. You can edit the following parameters in order to change clustering in this situation.

Note: ideally, your Data DRE should contain at least 500 documents.

SeedSize	Decrease **SeedSize** (by 3-4 points at a time). This reduces the size that seeds are required to reach, which means that more seeds are likely to be successfully created.
MinClusterDocs	Decrease **MinClusterDocs**. This means that clusters that contain fewer documents are not discarded.
StartingSuggestOverrideFactor	Increase **StartingSuggestOverrideFactor** (by 1 or 2 points only). This increases the number of sample documents from which Classification Server creates seeds, which in some cases increases the possibility of finding clusters in the data.
SeedBindLevel	Decrease **SeedBindLevel** (by 1 point at a time). This reduces the similarity threshold for clusters. You should not change this until you have tried changing **SeedSize**, since lowering **SeedBindLevel** is more likely to allow into clusters documents that are less relevant.

Large amount of data

If your DRE has a large amount of data, you will probably not need to edit any clustering settings - since this is the situation in which clustering is most successful. In some cases (for example, if your DRE contains more than 1 million documents), it may be beneficial to alter the following setting:

StartingSuggestOverrideFactor	Increase the value of **StartingSuggestOverrideFactor**. This increases the number of sample documents from which Classification Server creates seeds. This is sometimes necessary in order to allow a broader section of the DRE's content to be represented by the clusters that are created.

Very similar data

If the documents in your DRE contain highly similar concepts, then this may be reflected by Classification Server identifying a small number of large clusters. For example, if your DRE contains mostly documents about sport, then you may get one large "sports" cluster. This situation is a realistic characterization of the data in your DRE, but in many circumstances is not useful. You can edit the following settings in order to generate smaller, more specific clusters (for example, breaking "sports" into "football", "tennis", "golf"…):

SeedBindLevel	Increase **SeedBindLevel**. This requires greater similarity between the documents that form a seed, which can have the effect of reducing the breadth of topics covered by the concepts in the documents that a seed contains. **Note:** increase **SeedBindLevel** 1 point at a time; increasing by too much can result in seeds getting discarded because they don't contain enough documents.
BindLevel	Increase **BindLevel**. This requires greater similarity between the concepts in seeds or clusters that are merged to create a cluster, which can have the effect of decreasing the size of clusters, as well as increasing the number of clusters identified, because merging seeds and clusters together is stopped at an earlier stage.

Very different data

If the documents in your DRE contain a wide variety of concepts, there may not be enough similar documents for Classification Server to create seeds or clusters that characterize the data in the DRE. You can lower the similarity requirement with the following settings:

SeedBindLevel	Decrease **SeedBindLevel**. This reduces the similarity requirement between the documents that form a seed, which can have the effect of increasing the breadth of topics covered by the concepts in the documents that a seed contains. **Note:** decrease **SeedBindLevel** 1 point at a time; decreasing by too much can result in seeds and clusters containing documents that are less relevant, because the similarity requirement is too low.
BindLevel	Decrease **BindLevel**. This reduces the similarity requirement between the concepts in seeds or clusters that are merged to create a cluster, which can have the effect of increasing the size of clusters, as well as increasing the number of clusters identified (since fewer get discarded for not meeting the quality requirement).

Configuring clustering to change the data view

It might be the case that although Classification Server identifies clusters that characterize your data successfully, you want to change the view of the data that clustering creates. The following settings enable you to change the data view that clusters generate:

NumClusters	Increase **NumClusters** to get a more low-level view of your data by identifying more clusters. Decrease **NumClusters** to get a more high-level view by identifying fewer clusters.
MinClusterDocs	Decrease **MinClusterDocs** to reduce the number of clusters that are discarded. This allows smaller clusters to be identified. Increase **MinClusterDocs** to increase the number of clusters that are discarded. Only larger clusters are kept.

BindLevel	Decrease **BindLevel** to reduce the similarity requirement between the concepts in seeds or clusters that are merged to create a cluster. This can have the effect of increasing the size of clusters, as well as increasing the number of clusters identified (since fewer get discarded for not meeting the quality requirement). Increase **BindLevel** to increase the similarity requirement between the concepts in seeds or clusters that are merged to create a cluster. This can have the effect of decreasing the size of clusters, as well as increasing the number of clusters identified, since merging seeds and clusters together is stopped at an earlier stage.

12. Legacy compatibility

Importing categories from legacy topic sets

Classification Server allows you to build categories from topic sets from a legacy keyword engine. The **CategoryImportTopic** action creates one category per topic set. You can specify whether you want to maintain the original Boolean rules of the topic or import the topic as an Autonomy concept matching agent.

If you want to adjust the weightings of the terms for your category, you can use the **CategorySetTNW** action to edit the weights for existing terms or to add new terms and assign weights to them. (The **CategoryGetTNW** action allows you to view the current category terms and their weights.)

Indexing manually tagged documents into the DRE

Indexing XML-tagged documents

You can index XML-tagged documents directly into a DRE 4 / AXE using **DREADD**.

Enter the name (or full path) of your XML file in the following format:

http://<host>:<port>/DREADD?<file_name>

This indexes the fields in the XML file that you specify into the DRE.

Optional parameters for **DREADD** allow you to explicitly set the following details when you are indexing XML:

- document format (XML or IDX), if format is ambiguous
- fields that contain the document's date
- fields that contain the name of database to index into
- fields that contain the document's expiry date
- fields that contain delimiters for start and end of document
- fields to discard before indexing
- fields to flatten hierarchy of a structured document
- fields that contain the document's security type
- fields that contain the document's language
- fields that that mark the start of a new section of the document

Manually tagged documents

You can index non-XML format tagged documents into a DRE using the Autonomy Connectors and the Import module, which import documents to IDX format ready to be indexed into the DRE. When you index IDX documents into the DRE, meta data is automatically extracted from the documents' content. See the documentation for your Connector for details of how to set it up for your document format.

Legacy search

Boolean, keyword, wildcard and proximity searches are supported for queries to the DRE, as well as a number of advanced searches. See the **Querying the DRE** chapter for a list of the types of search that are supported, and refer to your DRE documentation for details of how to implement your search.

Glossary

ACI (Autonomy Content Infrastructure)

The Autonomy Content Infrastructure is a technology layer that automates operations on unstructured information for cross enterprise applications, thus enabling an automated and compatible business-to-business, peer-to-peer infrastructure.

The ACI allows enterprise applications to understand and process content that exists in unstructured formats, such as e-mail, Web pages, office documents, and Lotus Notes.

AXE (Autonomy XML Engine)

The Autonomy XML Engine (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.

Connector

A Connector is an Autonomy fetching solution (for example HTTPFetch, Oracle Fetch, AutoIndexer and so on) that allows you to retrieve information from any type of local or remote repository (for example, a database or a web site). It imports the fetched documents into IDX or XML file format and indexes them into a DRE from where you can retrieve them (for example by sending queries to the DRE).

Database

An Autonomy database is a 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 (for example, HTTPFetch, AutoIndexer, ODBC Fetch and so on), 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.

IAS (Intellectual Asset Protection System)

The Intellectual Asset Protection System provides an integrated security solution to protect your data. At the front end, authentication checks that users are allowed to access the system on which result data is displayed. At the back end, entitlement checking and authentication combine to ensure that query results only comprise documents that the user is allowed to see, from repositories that the user is allowed to access.

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 DRE 4 / AXE manual for details).

Importing

After a document has been downloaded from the location it is stored in, it is imported to an IDX file format. This process is called "importing".

Indexing

After documents have been imported to IDX file format, their content (or links to the original documents) is stored in a DRE. This process is called "indexing".

Permission

Depending on the roles that have been allocated to a user, he has permission to access specific portlets (this may include the ability to edit them).

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.

Role

Each user is allocated one or more roles by the administrator, which assign him to a certain group of users. The roles that a user has determine which privileges he has (that is which portlets he has access to).

Index

I

K

L

M

N

P

Q

R

S

T

X