

Saxonite Limited

Indexing Service

Best Practices

Author: Keith Young
Reference: ArtiSanIndexing/1
Date: 15 November 2005
Version: 1.0
Distribution: Company confidential

Abstract

This document discusses the Microsoft Indexing Service and describes some best practices for using it with ArtiSan.

© Saxonite Ltd 2005

Change History

1.0 Initial

Contents

Introduction	2
The Indexing Service	3
The Significance of the Change Journal and Scans	5
Catalog Names, Cycling Indexes and Documents to Index	6
Backing Up, Moving and Rebuilding the Catalog	7
Best Practices	8
Registry Configurations	9

Introduction

The main purpose of this document is to discuss the function of the Microsoft Index Server and to describe “best practice” methods that can be used with Index Server (IS) as part of an ArtiSan install.

Full-text indexing is a complex technology requiring the interaction of a large number of components operating in a fairly unpredictable environment (attempting to read and filter documents of unknown type). The potential for issues to arise is quite large.

The key requirements of the Indexing System are that it should be efficient, reliable and scaleable. Generally, the indexing system achieves efficiency by trying to avoid aggressive use of CPU and IO resources to allow other components in the system to run. It builds its indexes in the background making searches for content more efficient through the use of both a full-text index and a property index. Reliability and scalability can be greatly improved through the adoption of best practices when administering the service. Many of these best practices are described on the Microsoft Web Site (see <http://msdn.microsoft.com/library/en-us/dnanchor/html/indexserv.asp>).

However, all software has issues that may cause corruption to the data it manages. Best practice strategies attempt to identify procedures that help to avoid the corruption taking place or to ease recovery from it. Corruption of an index is a significant problem because it means that the index has to be completely rebuilt. This can be very expensive in a large repository. Our experiments on low spec server hardware seem to suggest that you can index at least 1-1.5 million documents a day, though the actual figure will vary depending on the underlying hardware employed and the amount of available CPU and I/O resource. Assuming this figure, if you have a work queue of 20 million documents it would take more than two weeks to process this queue.

The Indexing Service

The Indexing Service is supplied as part of the NT operating system to facilitate faster file searches and to enable full text indexing on web sites. The software was part of NT4 and has gone through 3 major releases; the main changes were to improve its reliability. The Indexing server reads files from a repository and indexes the text in the files and "properties" of those files. Properties might include things like the size of the file, the modification date, the author and so on. The Indexing Service provides an API (actually a number of them) through which the indexes can be searched either for text and/or for property values. Indexing Service also supplies an API for management and some other APIs designed to allow end users and ISVs to extend the service. More recently, MS has been moving over to using the MSSearch service (MSSearch is used in SQL Server, Exchange Server and so on.)

Essentially, Indexing Service is configured with one or more "Catalogs". Each Catalog is a set of files containing the index and property information as well as workflow and management information. Each Catalog has an associated set of directories (called Scopes) containing the data to index (the repository). This association is maintained in the system registry.

The Indexing service can use a combination of change journals and file system scanning to ingest content from the repository for indexing. As files are changed, the service reads the new version of the file that was changed and processes the file content.

The indexing process includes filtering (extracting text and properties), word-breaking and stemming. Indexing Service has an extensible architecture when it comes to the basic operations of parsing content. It supplies APIs to allow ISVs to:

- a) Add new filters (e.g. zip, PDF, etc.)
- b) Add new languages (e.g. Serbo-Croat, Mongolian)

The only important API that is missing is an API to "send" a document to the indexing system. Indexing Service only indexes from files in its repository.

If the filtering process fails for some reason, Indexing Service places the document on a secondary queue for retry later. The number and timing of retries is controlled in the registry and the information is displayed in the MMC UI as "Deferred for Indexing".

One problem with the Indexing Service is that the indexing process appears to be single-threaded. This is in accord with trying to limit the cost of indexing in CPU and IO terms, but is less effective when trying to index large repositories.

Once the system has a set of words and properties, it places these into a word list in memory (the words) and into the property cache (the properties). The indexing system grows the word lists to configured maximum size and then merges them into a shadow index on disk, optimising the space used. The shadow indexes grow over time (actually quite quickly). Once they achieve a given size, the shadow indexes are merged into a master index. These operations can cause significant sudden changes in size to both files in the system and memory usage. The properties are streamed to property storages on disk. The master and shadow indexes and property stores are used as the basis of queries.

When performing a query, Indexing Service filters result sets according to permissions of the searching user; eliminating documents that the user cannot access. However, this only works where the repository is on the same machine (local) as the Indexing Service. To get round this issue, we have supplied an add-on filter that extends the system to store permissions information as properties of the documents so that we can filter the results through queries on specific properties. The PermEML module also "fixes" some issues with the standard EML filter and foreign language text and adds some specialised properties relating to journals.

An important point to understand is that the Indexing Service places a lot of emphasis on being as discreet as possible taking every opportunity to back off and defer operations. When you consider its use as a way of speeding up file searches on a desktop machine,

this makes a lot of sense. However, when using the system as an integral part of an ArtiSan archive this makes less sense.

The Significance of the Change Journal and Scans

As stated above, the Indexing service can use a combination of change journals and file system scanning to ingest content for indexing. Normally, the Indexing Service reads file change information from the NT Change Journals associated with the repository. This is an operating system provided facility that records the details of each change to a file system in a circular fixed size buffer (on disk).

Index Server also has two types of scan named incremental and full. An incremental rescan checks standard properties of files (e.g. date, size, etc.) with that it has recorded for the file and if they are changed adds them to the list of documents to index. A full rescan simply adds all files in the repository to the documents to index. In fact, full rescans are a major cause of problem in larger repositories because of the time taken to do a scan of all files and directories in a large repository. You probably NEVER want to do a full rescan unless you are adding new filters to the system.

It is important to note the significance of the change journals particularly for larger repositories. Index Server does not physically scan for files and directories unless:

- a) It is told to do so through the management console or the Administrative objects
- b) It cannot keep up to date with changes in the change journal

This is important because a scan on a larger repository can take a very large amount of time as evidenced by the statistics above. A typical change journal size is about 8MB (the default). Each change entry is approximately 52 bytes in length allowing for a total number of journals to be 161K journals. Here is a section of the change journal for a typical system:

```
HA: FILE_CREATE
HA:
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml: FILE_CREATE
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml:
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml:
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml: SECURITY_CHANGE
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml:
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml: SECURITY_CHANGE
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml:
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml: SECURITY_CHANGE
IJDHADMNDNGMIJFEEDHMJFOKINGPPMHL_20050523.eml:
```

The first two entries are there because ArtiSan created a new directory call HA. The rest are ArtiSan creating a new archive item. As you can see, ArtiSan typically uses 9 entries for each archive operation and so the change journal might actually hold only 10K entries. Now, let's assume that the system is archiving 10 messages a second. The change journal would only cover 16 minutes. You can increase the length of the change journal (at the cost of disk space), but the operation requires that you delete and recreate the journal. Index Server records the last USN (a unique number for each change to the volume) and the Journal ID (a number NT invents when a change journal is created). Index Server then asks for all changes on a particular journal from a given USN. If you delete the journal and recreate it, the Journal IDs will not match and Index Server would force an incremental rescan. Generally, if you want to change the Journal length, you should do so at install time. Another point is that given the example above, you can only stop the Indexing Service for up to 16 minutes if you want to avoid a rescan because the journal will overflow. Incidentally, if there is no change log, Indexing Service will create one (using a registry entry to specify its size defaulting to 8MB).

A second consideration is that the fact of when adding permissions on a single instance item, the Indexing service will receive a change notification for each additional person who has a copy of a given mail. Indexing Service does not know that the file is simply a security change (actually it does, but it does not handle this in a very intelligent manner) and so will re-index the file as if it is new content (or had been modified).

The Indexing Service "documents to index" is not a queue that maintains unique references to files, i.e. the same file can appear more than once in the queue. Consequently, it is possible to have a queue that is larger than the number of files in the repository. Also, Indexing Service seems to count directories as well as files in many of its statistics.

Catalog Names, Cycling Indexes and Documents to Index

Another important point about the Indexing Service is that the Catalog name is not significant except in regard to the registry location for its configuration. For instance, you can create a duplicate of an existing index with a new name, populate that index with data and then simply rename the registry keys and use the new index (although you have to stop and start the indexing service between doing the rename).

The significance here is that it means that you can continue to use an existing index (which may be broken in some way) whilst building a new version and maintain the service to end-users throughout the operation.

Equally, suppose you have an index that has a large number of documents to index (probably because someone has accidentally selected “Rescan Full” or equivalent). One possible solution is simply to rebuild the catalog. However, the index would now be offline for a long time. A second possibility is to simply create a new catalog with a different name and then when completely indexed, rename the catalogs. The problem here is that the server will then be doing twice the work. However, since Indexing Service always keeps its CPU and IO requirements low, this should not be too much of a problem.

There is a third alternative. In this version you temporarily exclude the Scope from the Catalog. Indexing Service will remove the items from the index and the “Documents to Index” will drop to zero. In you then re-include the Scope, the system will rebuild the index. However, while the system is set to No and during the subsequent rebuilds, the search system will be either offline or incomplete, but the operation takes significantly less time to complete. On balance the best approach is to build a new alternate index.

Backing Up, Moving and Rebuilding the Catalog

Maintaining a regular and recent backup of an index can help where the index is corrupt because you can simply restore the backup and proceed from where the backup took place. The consequence would be an incremental rescan, but you would get back online much faster. Indexing Service backup is document at:

<http://support.microsoft.com/default.aspx?scid=kb;en-us:247093>

In some cases, you may need to move the Indexing Service catalog to a larger drive. The simplest way to achieve this is to delete the existing catalog and rebuild the new one. However, this means that searches would be offline whilst the operation completes and, as we have said, this could be a significant amount of time (potentially days). The alternative would be to backup and restore to the new location as documented above:

The important thing to understand about the index server catalog is that it is just a set of files. You can backup or copy the catalog (catalog.wci) using SCopy (described <http://support.microsoft.com/kb/q174273/>) and then make the index server point at the copy and it will work.

Typically the process is to:

- a) Shut down the Index Server
- b) Move or copy the catalog.wci to the new location (use SCopy to keep the permissions)
- c) Open regedit.exe and navigate to the ArtiSan catalog
- d) Edit the Location Value to point at the new location
- e) Restart the Index Server
- f) Test the search facility

In the unlikely event, that you need to completely rebuild the catalog, you should

- a) Stop the indexing service
- b) Using MMC, delete the ArtiSan catalogue (it will delete the relevant files)

Then you follow the same steps to recreate the catalogue as you did when installing it in the first place. This is either by:

- a) Using the ConfigureIS.vbs script from the bin directory or,
- b) Following the instructions given in the manual.

However, you should note that if you do this the index will be offline for a significant amount of time.

Best Practices

- 1) Always disable anti-virus and backup software on both the Catalog and the repository. Using AV and backup on a Catalog can cause a corrupt index due to unexpected file locks and deletions.
- 2) Never store a catalogue in a web virtual directory for the same reason.
- 3) Regular backup can help avoid the need for rebuilding the index in the event of a corruption. Backing up the indexing service catalogue is described above. It is important to note that:
 - a) You must always backup and restore permissions along with the files for both the Indexing Service Catalog and the Repository.
 - b) You must stop the indexing service as well as the scheduler service when doing backups (in v2, you can place the store in read-only mode to backup whilst data is still online).
- 3) Never issue the command "Rescan (Full)" on a large repository. Indexing Service is not very sophisticated when it comes to rescanning repositories. It simply adds all the items in the repository to the end of its outstanding queue (even if the item is already pending). Consequently, if you have a 5 million document repository and you issue this command twice, you will finish up with a queue with 10 million documents in it. This means the next document added to the repository will have to wait until the system has indexed the 14 million documents first, i.e. days. The only times you should issue this command is if you are adding a new filter to the filter set. If the index is corrupt, you will need to rebuild from scratch and you are better off starting with a new index.
- 4) Maintaining larger than standard Change Journals can improve the system by avoiding the need to rescan all files and folders. Essentially, if the Indexing Service runs out of change journal space, it will have to force a rescan resulting in large work queues.
- 5) Make sure you are not prioritising Application over Background services. By default, Indexing Service is always attempting to steal idle time to do its processing. If there is insufficient idle time, it will get behind, the change journals will overflow and you will be back to point 4).
- 6) Never encrypt the store directory. Indexing Service will not index encrypted files. However, you can compress the file system to get extra space, but this will slow down the indexing and accessing of data.
- 7) Where possible keep the indexing service on the machine where the store is (much lower network bandwidth). Indexing Service demands that the catalogue is local for the same reason. If this means that the index is remote from the ArtiSan web site, you will need to change the query.asp to use the remote index. Note also, that access to Indexing Service can be controlled by a GPO, so the configuration can be very tricky. If it is working, then it is best not to change this.
- 8) Always use an NTFS drive. Indexing Service is not able to honour permissions on a FAT or other format drive. Also FAT does not support change journals and so Indexing Service will be completely rescanning the file system every couple of hours, by default, creating enormous work queues.

Registry Configurations

You can improve the performance of the Indexing Service by changing the following values under the key:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ContentIndex
```

This registry key operates like the IIS metabase for many of the properties. You can specify properties for a given Catalog or across the entire service. Some properties only work at the service level.

Documentation for some, but not all, the registry entries can be found at:

http://msdn.microsoft.com/library/en-us/indexsrv/html/ixrefreg_0841.asp

Many of the registry entries relates to querying and use of the ISAPI search add-ons which we are not concerned with here.

CiCatalogFlags

This MUST be set to 0. Any other value may cause the IndexServer to force a rescan of documents rather than using the Change Journal.

DelayUsnReadOnLowResource

DelayUSNReadOnLowResource, ideally should be set to 0, but as the documentation points out, if low resource conditions exist the Indexing Service will back-off anyway, so there is probably no need to read the Change Journal because nothing will be indexed. The actual reading of the change journal is controlled by the entries USNLogAllocationDelta, USNReadMinSize, USNReadTimeout. Adjusting these values may help to achieve more reliability, but are going to make very little difference to the overall performance of the system.

MaxWordListIO

MaxWordListIODiskPerf

This should help to avoid the Indexing Service backing off under heavy IO load, but our experiments suggested it made very little difference

PropertyStoreMappedCache

SecPropertyStoreMappedCache.

We raised the values from 8 to 128 for both these properties and this increased the rate from 40K/hour to 65K/hour.

ThreadClassFilter

ThreadClassPriority

We tried this on our system setting ThreadClassFilter to 0x80 (HIGH) and ThreadClassPriority (2), but were disappointed at the results. Our system seemed to be constantly waiting on IO. This may be due to the slow disk drives we were using.

WordListUserIdle

This should always be set to 0. This is setup by default by the installation scripts.

IsIndexingW3Svc

IsIndexingNNTPSvc

W3SvcInstance
NNTPSvcInstance

These should all be set to 0. This is setup by default by the installation scripts.

FilterFilesWithUnknownExtensions

This should be set to 0 to avoid the Indexing system trying to index the old ART files. This is setup by default by the installation scripts.

GenerateCharacterization

This should be set to 1 to generate the textual summary of mails that is optional on the search page. This is setup by default by the installation scripts.

MaxCharacterization

By default this should be 320. This is setup by default by the installation scripts.