SYBASE®

Installation and Administration Guide

Sybase® Search

3.1

[Microsoft Windows 2000, 2003, XP, and Sun Solaris]

DOCUMENT ID: DC35131-01-0310-03

LAST REVISED: September 2006

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This guide is for Sybase® Search version 3.1 administrators and other professionals who are familiar with their system's environment, networks, disk resources, and media devices.

How to use this book

This book contains the following chapters:

- Chapter 1, "Retrieving Information Intelligently," introduces Sybase Search and describes the features and architecture of Sybase Search.
- Chapter 2, "Installing Sybase Search," describes how to install Sybase Search.
- Chapter 3, "Administering Sybase Search," describes how to administer Sybase Search.
- Chapter 4, "Configuring Sybase Search," describes the key configuration parameters for containers, modules, and the hub. It includes tips for using the configuration files and changing parameters
- Chapter 5, "Configuring Web Administration," describes the key configuration parameters for the Web application provided with Sybase Search.
- Chapter 6, "Customizing Sybase Search," provides information about developing custom filters, parsers, and text splitters.
- Appendix A, "Silent Install Mode,"describes how to use silent install mode to install Sybase Search.
- Appendix B, "Generated Files," gives the names and locations of generated module files.

Related documents

The following Sybase Search documents are available:

- The Sybase Search Version 3.1 *Release Bulletin* for Microsoft Windows 2000, 2003, XP and Sun Solaris, document number DC35130-01-0310-03

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

- **Finding the latest information on product certifications**

  1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

  2 Click Certification Report.

  3 In the Certification Report filter select a product, platform, and timeframe and then click Go.

  4 Click a Certification Report title to display the report.

- **Finding the latest information on component certifications**

  1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

  2 Either select the product family and product under Search by Base Product, or select the platform and product under Search by Platform.

  3 Select Search to display the availability and certification report for the selection.

- **Creating a personalized view of the Sybase Web site (including support pages)**

  Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

  1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

  2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

- **Finding the latest information on EBFs and software maintenance**

  1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

  2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.

  3 Select a product.

  4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The syntax conventions used in this manual are:

Key	Definition
commands and methods	Command names, command option names, utility names, utility flags, Java methods/classes/packages, and other keywords are in lowercase Arial font.
variable	Italic font indicates: • Program variables, such as *myServer* • Parts of input text that must be substituted; for example: *Server*.log • File names
File | Save	Menu names and menu items are displayed in plain text. The vertical bar shows you how to navigate menu selections. For example, File | Save indicates "select Save from the File menu."
`package 1`	Courier font indicates: • Information that you enter in a GUI interface, a command line, or as program text • Sample program fragments • Sample output fragments

<*installLocation*> refers to the Sybase Search installation directory; for example, *C:\Program Files\Sybase\Search3.1*.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Retrieving Information Intelligently

Sybase Search is a knowledge management system that automates the process of locating relevant business information within the masses of unstructured information stored in your organization's file systems, network drives, and databases. Sybase Search provides intelligent information retrieval, document management, and document categorization.

Sybase Search connects to file systems and databases. Using the content-based catalog and search tool, Sybase Search automatically analyzes, indexes, and categorizes data and prepares the system for users to visually navigate to the chosen category.

Managing unstructured information

In many organizations, employees keep their research, financial projections, and presentations on local PCs or team-shared space on the company network. Accessing such information – or even finding it – often proves difficult when staff or storage rules and structures change.

Sybase Search technology

Sybase Search technology extracts and processes the text content from file systems and databases where the content is unstructured. The ability to automatically process this content removes the need to index or describe information manually and allows organizations to automate such common business operations as data capture, retrieval, and linking. Sybase Search technology offers an efficient and cost-effective solution for searching unstructured information, regardless of the format and the language in which the content is written.

The Internet offers familiarity with keyword search, which is the most common type of search-and-retrieval technology. Most people are familiar with the process of retrieving the information by typing one or two relevant keywords into a search engine.

Keyword search technology requires a business to identify documents by associating keywords with the document, which are then used for subsequent retrieval. This process, known as document "tagging," can be costly and time-consuming.

Sybase Search provides the means to automatically capture and retrieve information based on concepts rather than keywords. Through the use of proprietary algorithms, Sybase Search delivers a language-independent product capable of operating without the costly overhead associated with tagging. This provides an essential tool for managing the proliferation of unstructured information in today's business environment.

Sybase Search features

Sybase Search offers a number of features that allow today's organizations to ease or eliminate the time and cost required to support the demands of managing an organization's unstructured information. These features include:

- The support of more than 250 different formats of data, including most types of document, presentation, spreadsheet, and Web content formats
- The automatic capture and aggregation all of unstructured data
- The elimination of preprocessing or manual tagging of files, greatly improving the accuracy and efficiency of document retrieval
- The extraction of paragraphs from matching documents
- The ability to find similar documents by automatically providing a set of relevant content that is conceptually related to each document
- The ability to scale to millions of documents using a fully distributed architecture
- The ability to query and process using natural language

- Language independence
- A well-defined Java and XML API that allows Sybase Search to be integrated easily into other applications

Understanding the architecture of Sybase Search

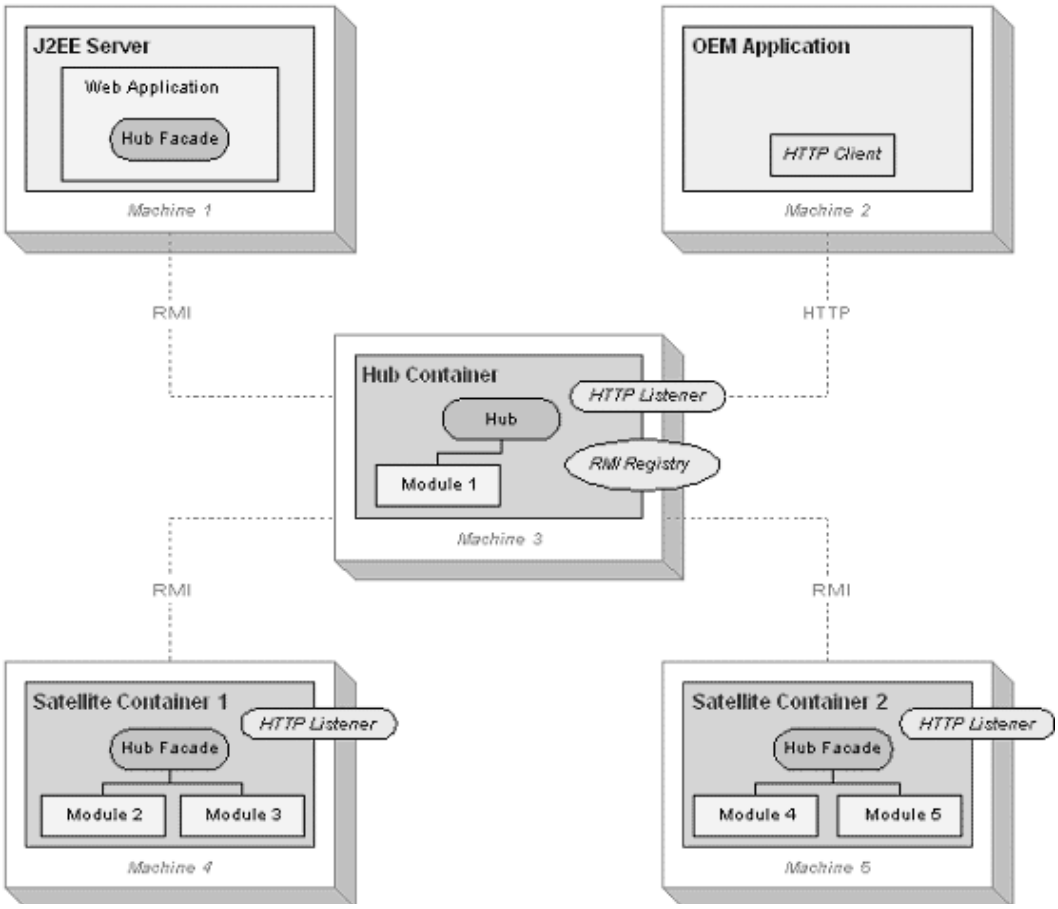
Sybase Search is a fully distributed system, with a central hub server and one or more satellite servers. Each server can contain one or many containers with one or more modules deployed in each container. The exact number of servers, containers, and modules depends on the needs of the Sybase Search installation.

The example architecture in Figure 1-1 contains:

- A central hub
- Two satellite containers
- A J2EE server containing the Web application
- OEM application connecting to Sybase Search

See Chapter 4, “Configuring Sybase Search,” for information about the various modules that comprise Sybase Search.

Figure 1-1: Example Sybase Search 3.1 architecture



Optimizing search strategies

As a **concept-based search engine**, Sybase Search performs best when you enter queries with search words in context (such as, in short phrases rather than as isolated words). In addition, if you know that more than one language is in use, repeating the concepts using different words generally improves results. Searching is often an iterative activity: you expand and refine queries based on the results returned.

Tips for optimizing search engine

This section provides tips for optimizing a concept-based search engine, which provides greater flexibility than traditional approaches to free-text searching, such as the Boolean combination of keywords.

For example, a user receives an e-mail message that says:

Following the incident close to Watford railway station in July, we need to assess the damage being done by tree branches tangling in overhead power lines or falling onto the tracks.

The user then wants to locate documents matching the e-mail message. Using a traditional search method, he or she might enter something similar to:

```
branches AND lines AND tracks
```

In this query, the user is using the Boolean operator "AND" to filter the information. This type of query is very precise and is helpful when:

- The user knows exactly what information is required, and it can be expressed in just a few words.
- There is no ambiguity in the words used in the query.
- The vocabulary of the target documents is known precisely.

In practice, this is rarely the case. It is more common that users are unsure of how to formulate their query precisely, thus introducing ambiguity within the query. Differing vocabulary used in documents to describe similar concepts can also result in important documents being missed altogether and too many irrelevant documents being returned.

If the user is searching a large database of documents, a query like the one in the previous example may retrieve a large number of items, many of which are not relevant to the specific query due to the search for a small number of specific, isolated words. Words like "branches" and "lines" are ambiguous and are common in a database of documentation concerning the railway system.

Query a number of concepts

Sybase Search is better suited to a query that contains a number of concepts and is expressed using ambiguous language, thus increasing the likelihood that the user retrieves results that are relevant to the query.

Using the previous e-mail example, isolate the key concepts, which are:

- Damage being done by tree branches
- Tangling of overhead power lines
- Falling trees and tree branches
- Obstruction or damage to tracks

Irrelevant concepts might include:

- Watford Railway Station
- July

Inclusion of irrelevant concepts distorts the search and may introduce some unwanted documents. So, a more effective query is:

```
damage being done by tree branches, tangling of overhead
power lines, falling tree branches, obstruction and
damage to tracks
```

Note You do not need to delimit concepts using a comma.

This is a better query because it contains all of the key concepts in the original query and expresses them using words in context. Results returned by this query are likely to produce significantly better results than the first attempt.

Adding variations

However, it is likely that some relevant documents will still be missed, due to differing vocabulary. Therefore, if you use your knowledge of the environment and expand the original concepts to include variations that you know from experience tend to occur, this may produce a query similar to:

```
damage being done by tree branches, tangling of overhead
power lines, falling tree branches, obstruction and
damage to tracks, forestry, wind damage, storm damage,
damage to rails, lines being pulled down by trees blown
over
```

At first, this may seem more confusing and less precise than the previous examples, but in fact it contains additional ways of defining the original concepts. You may find that no documents achieve a 100% relevance score with this query because no document includes all of these combinations. However, the most relevant documents are at the top of the list.

Often, you can improve search results by feeding back information from documents discovered by the system. For example, if a search produces a document that is relevant but the terminology used in the extracted summary is different from the search text, you may want to expand the original query by appending words or phrases from the document search results. In this way, the search becomes more accurate as you provide additional information.

CHAPTER 2

Installing Sybase Search

Preparing to install Sybase Search

Before installing Sybase Search:

- Verify that you have write permission on the directories where you plan to install the Sybase Search software.
- Make sure each server in the Sybase Search distributed environment meets the recommended or minimum requirements summarized in Table 2-1.

Table 2-1: System requirements

Operating system	Release level	RAM	Disk space
Microsoft Windows 2000, 2003, and XP 1GHZ minimum 2GHZ recommended	Any	512MB 1GB or more recommended	100MB for system files 1GB for indexed data
Sun Solaris 64-bit (SPARC)	OS 9 and above	512MB 1GB or more recommended	100MB for system files 1GB for indexed data

Note See the Technical Library Product Manuals Web site at http://www.sybase.com/support/manuals, or see the release bulletin for components that require operating system patches.

Determining hardware requirements

Before installing Sybase Search, you should decide what hardware is required. Consider the following questions when determining hardware requirements:

- Approximately how many documents are likely to be indexed?
- Approximately how many users are likely to access the system concurrently?
- What are your performance targets?

Depending on the hardware specifications, Sybase recommends that you plan for one server per 500,000 documents indexed, with an additional server for the Sybase Search Hub. For example, estimating 2 million documents to be indexed would require five servers. For smaller installations, it may be sufficient to install both the Hub and other modules on one server.

Determining a deployment strategy

After you determine the hardware configuration, determine how to deploy Sybase Search across multiple servers. Ideally, there should be no more than one container per server. Multiple containers residing on a server must access the same disk drive, which can slow down performance. The Java 2 Platform Enterprise Edition (J2EE) server hosting the Sybase Search Web application should also reside on its own server. Distributing Sybase Search across multiple servers helps maximize the resources available to each container and helps prevent I/O bottlenecks.

Installing one container per server also reduces unnecessary network traffic among containers. If two containers are installed on one server, their network traffic can be eliminated by combining their internal modules together into a single container. There is no advantage to running more than one container on a single server; it is not recommended.

For a small installation on one server only, Sybase recommends that only one container is used, with the modules often shared across multiple containers located together in a single container.

Note Each container runs within a Java Virtual Machine (JVM) tied to a single CPU. It is possible to run multiple containers on a server with multiple CPUs, with each container's JVM attached to a different CPU.

A clear understanding of performance expectations and how many documents you plan to index help determine how many servers are needed in your environment. Table 2-2 shows an example.

Table 2-2: Example of server requirements

Setup	Sybase Search feature	Number of servers
Small: Less than 500,000 documents are planned to be indexed.	Single server installation	One server
Medium: 500,000 to 1.5 million documents are planned to be indexed.	Hub container Satellite container Web application	One server One server One server
Large: More than 1.5 million documents are planned to be indexed.	Hub container Satellite container Web application	One server Five servers One server

Determining module groups

Sybase recommends grouping the following modules in the hub container:

- Unique ID (UID) Generator
- Document Group Manager
- Text Manager
- Term Lexicon Manager
- Metadata Manager
- Query Manager
- Repository Manager
- Schedule Manager
- Category Manager

Sybase recommends grouping following modules in each satellite container:

- Term Lexicon Manager Delegate
- Metadata Manager Delegate
- Filter Factory Manager
- File System Document Store Module

- Database Document Store Manager

Installing Sybase Search

The Sybase Search Universal Installer consists of platform-specific executable files so that you can install Sybase Search on the following operating systems:

- Microsoft Windows 2000, 2003, and XP
- Sun Solaris

The installer runs in interactive mode and silent mode:

- **Interactive install mode** requires you to run the appropriate platform-specific executable. The installation process consists of a series of interactive dialog boxes in which you enter parameter settings.
- **Silent install mode** is used primarily with the Sybase Data Integration Suite. You can also use silent mode to install Sybase Search if you only have command line access to a server. For more information about silent install mode, see Appendix A, “Silent Install Mode.”

Each installation type installs a Sybase Search feature or combination of features. Typically, you install only one feature per server; therefore, you run the Sybase Search installer for each server. The Sybase Search installer provides the installation types summarized in Table 2-3 on page 11.

Table 2-3: Sybase Search installation types

Installation type	Description
Typical	This installation type installs all the components for running Sybase Search on one machine. The components include a single container, which consists of all the Hub and Satellite modules in one container and the Web Administration server.
Custom	This installation type lets you install one or more of the following components: • Hub container - installs the Hub container, which runs the central hub of Sybase Search and is used to coordinate all other satellite containers. • Satellite container - installs a satellite container, which is a remote component of Sybase Search and contains the modules required to distribute the indexing and search modules. • Web Administration server - installs Hyena, which is a lightweight J2EE-compliant JSP/servlet container and the Web application for performing Sybase Search administration tasks.

You can install the features separately or install combinations of features. For example, the Typical installation type installs all the components necessary for running Sybase Search on a single server. The Custom installation type allows for combinations of a hub container, satellite container, and Web Administration server to be installed on one server.

❖ **To install Sybase Search in interactive install mode**

1 Verify that you have write permission to the directories on each server where you are installing Sybase Search features.

2 Run the interactive installation program.

- If you are installing on Windows 2000, 2003, or XP, run OmniQ_3.1_win32.exe.
- If you are installing on Solaris, run OmniQ_3.1_sunsparc64.bin.

The Welcome page appears.

3 From the Welcome page, click Next. The License Agreement page appears.

4 Accept the conditions of the license agreement and click Next. The Install Directory page appears.

5 Click Next to accept the default directory, or click Browse to select a different directory.

6 Select the installation type that best suits your requirements and click Next. (Table 2-3 describes the installation types.) The interactive dialog boxes are displayed.

7 Enter values in each field. Table 2-4 describes each field. The installer displays only dialog boxes containing fields relevant to the chosen installation type.

Table 2-4: Interactive install mode fields

Field name	Description
Container ID	Enter a value from `1` to `99` to uniquely identify a container. The Container ID of the Hub container is always set to `1`.
Container Port Number	Enter the number of the TCP/IP port on which the container's embedded HTTP server listens. The container's HTTP listener binds to the port number 8000 plus the container's ID. For example, the port is 8001 for container 1 and 8002 for container 2. No two containers can share the same port number.
RMI Port Number	Enter the port number where the remote method invocation (RMI) service runs. The port number that you enter for the Hub container must also be the port number entered for satellite containers. The Hub container default port number is 7000.
Web Administration Server Port Number	Enter the TCP/IP port on which the Web Administration server listens for connections.
Hub Container IP Address	Enter the IP address of the server on which the Hub Container is installed. For a single-server installation, the default is 127.0.0.1.

After you enter values and click Next, the Summary Information page appears.

8 Review the Summary Information page. If you are satisfied with the settings, click Install.

9 Repeat this procedure for each server in your Sybase Search environment.

Installation directories

The following directories are created for each installation type:

- *_jvm* – JVM bundled with the installer

- *_uninst* – Uninstaller files
- *jre* – JRE used by Sybase Search containers and servlet/JSP container
- *OmniQ* – Folder that contains configuration and application JAR files
- *Hyena* – Sybase Search servlet/JSP container used to run the Web administration pages
- *sx* – Stellent filters used to filter documents
- *webapp* – Sybase Search Web administration files

Table 2-5 on page 14 lists the folders created for each installation type.

Table 2-5: Installation folders

Installation type	Directories
Hub container	<*installLocation*>/_jvm <*installLocation*>/_uninst <*installLocation*>/jre <*installLocation*>/OmniQ/bin <*installLocation*>/OmniQ/config <*installLocation*>/OmniQ/config/dtd <*installLocation*>/OmniQ/data <*installLocation*>/OmniQ/lib
Satellite container	<*installLocation*>/_jvm <*installLocation*>/_uninst <*installLocation*>/jre <*installLocation*>/OmniQ/bin <*installLocation*>/OmniQ/config <*installLocation*>/OmniQ/config/dtd <*installLocation*>/OmniQ/data <*installLocation*>/OmniQ/lib <*installLocation*>/sx
Web Administration server	<*installLocation*>/_jvm <*installLocation*>/_uninst <*installLocation*>/jre <*installLocation*>/Hyena/bin <*installLocation*>/Hyena/config <*installLocation*>/Hyena/lib <*installLocation*>/Hyena/logs <*installLocation*>/Hyena/work <*installLocation*>/Hyena/work/omniq <*installLocation*>/webapp <*installLocation*>/webapp/image <*installLocation*>/webapp/script <*installLocation*>/webapp/WEB-INF <*installLocation*>/webapp/WEB-INF/lib

Installation type	Directories
Single server	<*installLocation*>/_jvm <*installLocation*>/_uninst <*installLocation*>/jre <*installLocation*>/OmniQ/bin <*installLocation*>/OmniQ/config <*installLocation*>/OmniQ/config/dtd <*installLocation*>/OmniQ/data <*installLocation*>/OmniQ/lib <*installLocation*>/Hyena/bin <*installLocation*>/Hyena/config <*installLocation*>/Hyena/lib <*installLocation*>/Hyena/logs <*installLocation*>/Hyena/work <*installLocation*>/Hyena/work/omniq <*installLocation*>/sx <*installLocation*>/webapp <*installLocation*>/webapp/image <*installLocation*>/webapp/script <*installLocation*>/webapp/WEB-INF <*installLocation*>/webapp/WEB-INF/lib

Starting and stopping Sybase Search on Windows

In Windows, you can start and stop Sybase Search containers and the Web Administration server from the Windows Start Menu or from a command prompt.

❖ **To start or stop Sybase Search from the Windows Start Menu**

1 From the Windows Start Menu, select Programs | Sybase.

2 Select Sybase Search 3.1.

- To start a single server installation, select Start Single Container. The single container and Web Administration server start and run in a Windows console.
- To start a hub container, select Start Hub Container. The hub container starts and runs in a Windows console.
- To start a satellite container, select Start Satellite Container *n*, where *n* is the number that identifies the satellite container. The satellite container starts and runs in a Windows console.

- To start the Web Administration server, select Start Web Administration Server. The Web Administration server starts and runs in a Windows console.
- To stop a single server installation, select Stop Single Container. The single container and Web Administration server stop and the Windows console closes.
- To stop a hub container, select Stop Hub Container. The hub container stops and the Windows console closes.
- To stop a satellite container, select Stop Container *n*, where *n* is the number that identifies the satellite container. The satellite container stops and the Windows console closes.
- To stop the Web Administration server, select Stop Web Administration Server. The Web Administration server stops and the Windows console closes.

❖ **To start or stop a Sybase Search container from a Windows command prompt**

1 Open a Windows command prompt.

2 Change directories to *<installLocation>\OmniQ\bin.*

- To start a container, enter OmniQEnterprise.bat -start *[ID]*, where *[ID]* is the container ID. The container starts and runs in a Windows console.
- To start the Web administration server, change directories to *\Hyena\bin* and enter Hyena.bat -start.
- To stop a container, enter OmniQEnterprise.bat -stop *[ID]*, where *[ID]* is the container ID. The container stops and the Windows console closes.
- To stop the Web administration server, change directories to *\Hyena\bin* and enter Hyena.bat -stop.

Starting and stopping Sybase Search on Solaris

For a Solaris operating environment, you start and stop Sybase Search containers and the Web Administration server from a UNIX command line.

Note Before starting any containers, you must set the environment variable *LD_LIBRARY_PATH* for the current profile so that the document filters operate correctly. To set this variable, run *<installLocation>*/OmniQ/bin/env.sh.

- To start a Sybase Search container, enter *<installLocation>*/OmniQ/bin/OmniQEnterprise.sh start *[ID]*, where *[ID]* is the container ID. For the Hub container, the ID is 1.
- To stop a Sybase Search container, enter *<installLocation>*/OmniQ/bin/OmniQEnterprise.sh stop *[ID]*.
- To start the Web Administration server, enter *<installLocation>*/Hyena/bin/Hyena.sh start.
- To stop the Web Administration server, enter *<installLocation>*/Hyena/bin/Hyena.sh stop.

Performing post-installation tasks

The internal data structures of Sybase Search rely on certain settings remaining unchanged. Therefore, before you start Sybase Search for the first time and before any indexing is performed, you must perform:

- Language configuration – you must decide which language Sybase Search uses as the default language for indexing and querying before starting Sybase Search.
  - If Sybase Search is required to work across multiple languages, no stop-words or word stemming are required.
  - If Sybase Search is required to work with one language only, that language must be reflected in the various Text Module settings. See “Setting Text Manager parameters” on page 51.

    If the language is not English, it may be necessary to write a new stemmer. For details on how to write and plug in a new language stemmer into Sybase Search, contact Sybase support.

For information on the language-specific configuration of various modules, see “Optimizing Sybase Search” on page 59.

- Hub configuration – base this on the level of stress and load the system is expected to cope with. This includes ensuring the various module caches are set to a high enough level, where RAM is available.

  For example, if the Query Module is located on the Hub, then its settings need to be reviewed to ensure that it can handle the required number of concurrent queries.

  For details on configuring various Hub-specific modules, see “Configuring modules” on page 50.

- Remote container configuration – base this on the level of stress and load the system is expected to cope with. This includes ensuring that the various Module caches are set to a high enough level, where RAM is available.

  For more details on configuring various remote container modules, see “Configuring modules” on page 50 and “Configuring modules using system parameters” on page 68.

Installing Sybase Search as a Windows service

You can install and run any Sybase Search container as a Windows service.

1 From a Windows command prompt, change directories: `<installLocation>\OmniQ\bin`.

2 Enter the following command:
OmniQEnterprise.bat -install *[ID]*, where *[ID]* is the container ID. The hub container is always 1.

  This installs the “Sybase Search – Container [ID]”. Then, you can use the Microsoft Management Console to run the Sybase Search container as a Windows service.

  You can also run and install the Web Administration server as a Windows service:

3 From a Windows command prompt, change directories to *<installLocation>\Hyena\bin*.

4 Enter the following command:
```
Hyena.bat -install
```

This installs the service named "Sybase Search – Web Admin Server." Then, you can use the Microsoft Management Console to run Sybase Search Web Administration server as a Windows service.

Uninstalling Sybase Search

To uninstall Sybase Search, you can select which features to remove. Keep in mind that the following components are shared among features:

- JRE is shared by all features installed on a server.
- Stellent filters are shared by all satellite containers and single containers installed on a server.

The uninstall process removes all folders, including the configuration, data, and log directories. If you anticipate requiring configuration settings, data, or log files for further use or future installations, create backups of the following directories:

- *<installLocation>/OmniQ/config*
- *<installLocation>/OmniQ/data*
- *<installLocation>/Hyena/logs*

❖ **To uninstall Sybase Search from Windows**

1 From the Windows Start Menu, select Programs | Sybase.

2 Select Sybase Search | Uninstall Sybase Search. The Welcome page appears.

3 Click Next. The list of features is displayed.

4 Select the features that you want to uninstall and click Next. The Summary Information page appears.

5 Review the Summary Information page. If you are satisfied with the details, click Uninstall. The uninstall program removes the Sybase Search components.

6 Click Finish.

❖ **To uninstall Sybase Search from Solaris**

1 From a UNIX command line, enter the following command:

<installLocation>/_uninst/uninstaller.bin

The Welcome page appears.

2 Click Next. The list of features is displayed.

3 Select the features that you want to uninstall and click Next. The Summary Information page appears.

4 Review the Summary Information page. If you are satisfied with the details, click Uninstall. The uninstall program removes the Sybase Search components.

5 Click Finish.

Uninstalling the Windows service

If you installed a Sybase Search container as a Windows service, you must invoke *OmniQEnterprise.bat* with the uninstall parameter and the container ID for the service to be removed.

For example:
<installLocation>\OmniQ\bin\OmniQEnterprise.bat -uninstall [ID]

If you installed the Sybase Search Web Administration server as a Windows service, you must invoke *Hyena.bat* with the uninstall parameter for the service to be removed.

For example:
<installLocation>\Hyena\bin\Hyena.bat -uninstall

CHAPTER 3

Administering Sybase Search

Accessing administration pages

Sybase Search is administered through a J2EE Web application; therefore, you can administer Sybase Search from any machine that runs a Web browser. From the Sybase Search administration pages, you can view the distributed Sybase Search installation and administer it.

To access the Sybase Search administration pages, open a Web browser and enter the URL http://*hostname:port*/omniq. The *hostname* is the name or IP address of the machine hosting the Sybase Search Web application. The *port* value is the port number for the J2EE application server hosting the Sybase Search Web application. The default port address is 8081.

The Sybase Search administration pages consist of a Home page and the following pages:

- Search – shows the Search page and lets you search across all of the documents that you have indexed in Sybase Search.
- System – lets you view the distributed setup. From the System page, you can view environment details, memory usage, and events for all containers within the Sybase Search installation. You can also schedule tasks.

- Document Management – lets you add, update, and remove documents from Sybase Search indexes. You can also create and manage document stores, organize document stores into groups, and create document categories.

Searching across documents

From the Search page, you can search across all of the documents that have been indexed in Sybase Search. The Search page accepts the following search criteria:

- Search Terms – enter a natural-language query in the Search Terms field. The more information you provide, the more accurate your results are. See "Optimizing search strategies" on page 4 for more information.
- Not Terms – enter terms to indicate concepts dissimilar to those for which you are searching. Unlike the Boolean NOT operator, documents that contain the Not Terms are considered for retrieval. However, the number of Not Terms a document contains is considered by the scoring algorithm and its relevance score is downgraded accordingly based on the weight of the Not Terms that it contains.

  For example, a search for "operating systems" with Not Terms "Windows XP," would not discount a document for containing the phrase, "opens in a new window."
- Categories – create categories to group documents by content, independent of location or type of document store. You can then use categories to filter search results. You can also view lists of documents for each category. See "Categorizing documents" on page 41 for more information.
- Document Groups – limit your search to one or more predefined document groups by selecting specific groups from the Document Groups list. Only documents from the chosen document groups are included in the search results.

- Metadata – select from a list of predefined metadata parameters to include metadata in the search. Sybase Search supports text, integer, and date metadata types.

  Note Some metadata parameters are document-specific. For example, a Microsoft Word document can have a Word Count, whereas a plain text document cannot and an HTML document most likely does not. Metadata parameters that are guaranteed to be searchable for all documents are described as being reliable. When the parameter searched on is not supported or not present in a document, it is automatically excluded from the results.

  The predefined metadata parameters are primarily to be used in conjunction with the Search Term and Not Term search criteria to refine the search, but you can also use them independently for a metadata search. Search results from a pure metadata search have no meaningful relevance scores.

  You select an operator for each metadata parameter. All metadata types support the equal to (=) operator. The integer and date types also support greater than or equal to(>=) and the less than or equal to (<=) operators.

  You also enter a value for each metadata field that you define. The values for text types are processed as search text. In other words, search terms are stemmed and augmented through synonyms and acronyms (except file path fields). The numeric type values are numbers and the date type values should be in the format configured by the Sybase Search system administrator, for example `dd/mm/yyyy`. See “Configuring parsers to query metadata fields” on page 61.

  Table 3-1 on page 24 lists the predefined metadata parameters and types.

Table 3-1: Predefined metadata parameters

Name	Type	Reliable
Author	TEXT	No
Character Count	INT	No
Client	TEXT	No
Comment	TEXT	No
Company	TEXT	No
Creation Date	TEXT	No
Document Name	TEXT	Yes
Document Origin	TEXT	Yes
Document Path	TEXT	Yes
Document Size (KB)	INT	Yes
Document Type	TEXT	No
Editor	TEXT	No
File Type	TEXT	Yes
Keywords	TEXT	No
Language	TEXT	No
Last Modified	DATE	Yes
Page Count	INT	No
Project	TEXT	No
Publisher	TEXT	No
Reference	TEXT	No
Second Author	TEXT	No
Status	TEXT	No
Subject	TEXT	No
Title	TEXT	No
Word Count	INT	No

- Metadata Combination Operators – you can select two combination operators:
  - Within Expression – use the Within Expression operator when there is at least one metadata parameter with a value that consists of more than one term. When you set the operator to AND, every term must be present in the document metadata for the match to succeed. When you set the operator to OR, only one of the terms must be present in the document metadata for the match to succeed.

For example, when the metadata parameter is `Author = "John Smith"`, the Within Expression operator differentiates the two possible interpretations, which are `Author = "John AND Smith"` or `Author = "John OR Smith"`.

Note Sybase Search supports only one Within Expression operator, so you cannot perform a metadata search for `Author = "John AND (Smith OR Roberts)"`. However, Sybase Search processes each Equals expression individually; therefore, you can achieve the same effect by using two separate expressions and using the OR Within Expression operator and the AND Across Expression operator. For example, `Author = "John" AND Author = "Smith, Roberts"` returns documents only authored by John Smith or John Roberts.

- Across Expressions – use the Across Expressions operator when you have defined at least two metadata parameters. When you set the operator to AND, both metadata parameters must succeed for the match to succeed. When you set the operator to OR, only one of the metadata parameters must succeed.

  For example, when the metadata parameters are `Author = "Smith,"` `Title = "Algebra,"` the Across Expressions operator differentiates the two possible interpretations as:

  - `Author = "Smith" AND Title = "Algebra"`
  - `Author = "Smith" OR Title = "Algebra"`

  Note Sybase Search supports only one Across Expressions operator, so you cannot perform a metadata search for multiple Across Expressions operators.

- Result Options – further refine your search results by defining values for the following options:
  - Minimum Document Relevance – define the minimum relevance ranking that a document must score for it to be included within the search results. Documents with scores lower than the percentage that you enter are not returned.
  - Number of Results – define the number of document results to display for each page by selecting a value from the Number of Results list.

- Number of Paragraphs – define the number of document paragraphs to display for each result document by selecting a value from the Number of Paragraphs list.
- Score Unknown Terms – include unknown terms by selecting the Score Unknown Terms check box. When selected, terms unknown to the system (and therefore, do not exist in any indexed document) are considered by the scoring algorithm.
- Term Highlighting – specify whether to highlight terms in the search results by selecting the Term Highlighting check box. When selected, terms from the query are highlighted in the result paragraphs and in the plain-text versions of the matching documents, as shown by the view text links.

Tracking system details

From the System page, you can track the system details of each Sybase Search container from the following pages:

- Environment – the Environment page lets you view the following details about each container:
  - Host name and port on which each container runs
  - Loaded modules
  - Data and configuration directories

  You can also view the Java system properties of each container's Java Runtime Environment (JRE).
- Memory Usage – the Memory Usage page lets you track the memory consumption of the Sybase Search containers, including the JVM allocation and consumption. You can also track the resources loaded within the loaded modules, such as data caches.
- Events – the Events page lets you view pages of recorded events that have occurred within the distributed Sybase Search installation. Sybase Search records information, warning, and error events through a Hub Manager. A Hub Manager is always present within a participating Sybase Search container. Events can be selected by Hub Manager (container), filtered by type, returned, and sorted in chronological or reverse order.

- Scheduler – the Scheduler page lets you set up specific tasks to run at configured intervals, thus automating repeatable duties, such as index updates and unifications. The Scheduler is implemented as a module and resides on the container of the Sybase Search Administrator's choice. A single Scheduler manages the scheduled tasks for an entire Sybase Search deployment. See "Scheduling tasks" for more information.

Scheduling tasks

From the System page, you can select Scheduler to configure tasks that run at scheduled time intervals. Tasks can be added and edited, as necessary. Sybase Search displays all scheduled tasks in the Scheduled Tasks list. The Scheduled Tasks list shows when the task is scheduled to run, when it last ran, and how many times it has run.

You can set up the following Sybase Search task types:

- Log Janitor – examines the current logs and deletes old log files. Optionally, it can compress inactive log files.

  Note Deleted logs cannot be recovered. Log Janitor does not decompress logs it has already compressed.

- Document Store Runner – runs a full update on the document store at the configured interval.
- Index Unifier – unifies the document store's indexes at the configured interval. See "Unifying an index stripe" on page 38.

❖ **To schedule tasks**

1 From the System page, select Scheduler. The Scheduled Tasks page appears.

2 Select New Scheduled Tasks. The New Tasks page appears.

3 In the Label field, enter a phrase or short description to identify the task.

4 From the Type list, select a task type. Sybase Search displays fields relevant to the chosen task type. Table 3-2 lists the fields associated with each task type.

Table 3-2: Scheduled task types and associated fields

Task Type	Fields
Log Janitor	Container – select a container ID. Keep daily logs for – select the amount of time that you want the system to keep daily logs. Compress non-active logs – specify whether you want Sybase Search to compress non-active logs.
Document Store Runner Index Unifier	Document Store – select a document store.

5 In the Every field, enter how often you want the task to run. Your choices are:

- Never
- Minutes
- Hours
- Days
- Months

6 Click Create. You return to the Scheduled Tasks list. The new task is added to the list and runs at its scheduled interval.

Managing documents

From the Document Management page, you can add, update, and remove documents from Sybase Search indexes. You can add documents to document stores, create document stores in document store managers, and organize document stores into groups. You can also create document categories.

Creating document stores

A **document store** is a collection of documents in Sybase Search related by physical location. You can organize documents into the following types of document stores:

- File system document stores

- Database document stores

File system document stores

A **file system document store** represents one or more collections of documents imported into Sybase Search from a local file system, including mapped network drives. The file system document store accepts one or more directory roots (for example, *D:\documents\office*), the contents of which Sybase Search indexes.

Although documents from different file systems (*C:\docs* and *\\network-share\docs*) can coexist in the same document store, internally, all documents found in all root directories of a file system document store are indexed together. This means they share the same data structures, and they are updated and removed together. Sybase Search analyzes folders and subdirectories. Files with valid MIME types are then indexed. You can customize the list of valid MIME types.

Database document stores

A **database document store** represents a collection of documents imported into Sybase Search from one or more database tables. You use a SQL query to import documents from database tables into Sybase Search. (See "Constructing an import query SQL statement" on page 32.)

All data conversions are handled internally, including files stored in binary format and links to files elsewhere on a system. Sybase Search can import data from any database for which JDBC drivers can be obtained.

❖ **To create a document store**

1 Click Document Management. The Document Store Managers page appears.

2 From the Document Store Managers page, click Document Stores.

3 From the Document Stores page, determine the type of document store that you want to create.

- To create a file system document store, click Import from File System.
- To create a database document store, click Import from Database.

The Create Document Store page appears. The type of document store you selected is displayed in the Type field.

4 Complete the following fields:

Field	Description
Fields common to file system and database document stores	
Name	Indicates the name of the document store.

Field	Description
Manager	Indicates the document store manager for which the document store should exist. A document store manager manages zero or more document stores. Typically, there is one document store manager for each server where document indexing occurs. The document store manager for each document store that you create lets you set up document indexing on the different servers in the system. See "Managing document stores" on page 39.
Member of	Indicates the document groups in which the document store is a member. See "Grouping document stores" on page 40.
Not a Member of	Indicates the document groups of which the document store is not a member.
Index Now	Indicates whether to proceed with indexing immediately or to save the configuration without indexing at this time. See "Indexing document stores" on page 34.
Fields for file system document stores	
Directories	Indicates one or more root directory whose contents will be indexed and available for searching.
File Type Filter	Includes or excludes documents by file extension or MIME type, for example: • Include text/html – indexes only HTML documents. • Include doc – indexes only Microsoft Word documents. • Exclude text/xml, txt – indexes all documents except XML and text documents.
Fields for database document stores	
Host	Indicates the network name or IP address of the database server.
DB Name	Indicates the name of the database.
Username	Indicates the name of the user and authenticates access to the database.
Password	Indicates the password used to authenticate access to the database.

Field	Description
Presets	Indicates the type of database and the configuration of the Java Database Connectivity (JDBC) options. When you select a database from the Presets list, Sybase Search automatically displays the port, driver, and URL with common values for the type of database selected. The Presets list is configurable. To use a preset: 1 Complete the Name, Manager, and Member of fields for the database document store. 2 Complete the Host, DB Name, Username, Password, and Port fields for the JDBC connection details. 3 Select the a preset from the Presets list. The port, driver, and URL fields display the corresponding default values. 4 Click the Translate URL placeholders link to replace the URL template placeholders with the correct values. **Note** Inclusion of a database driver in the Presets list does not mean the driver is available to the system. Ensure that the driver is available to the selected document store manager.
Port	Indicates the port on which the database server listens.
Driver	Indicates the full class name of the JDBC driver.
URL	Indicates the JDBC URL to use to contact the database.
SQL Query	Indicates the SQL statement designed to import documents from a database. (See “Constructing an import query SQL statement” on page 32.)
Class	Identifies the document reference class, signifies the java class type that should be used by Sybase Search internally to store the DOC_REF SQL datatype. The document reference class is automatically determined the first time data is extracted from the database and it cannot be changed.
Length	Identifies the document reference length. The document reference length is only used for java.lang.String document reference types (the lengths of other types are implicit). In most cases, it should be the same as the VARCHAR column width from which the document references are being extracted. If the document reference is not a string, this value is ignored.

5 Click Create.

The document store is created. You return to the Document Store Information page of the document store that you created. The details of the document store display its number of searchable documents and a list of the indexed document roots. An indexing summary is also listed and, if the store is being indexed, the current indexing session information is displayed. See "Indexing document stores" on page 34 for more information about Sybase Search indexing concepts.

6 Click Edit to edit the attributes of the document store.

You can edit most of the document store attributes. For example, you can rename a document store; add or remove document roots; add or remove File Type Filters; and move the document store in and out of document groups.

7 Click Remove to remove a document store.

When you remove a document store, all settings and indexes are permanently removed from the disk. All documents indexed under the removed document store are no longer returned in searches.

Constructing an import query SQL statement

You construct a SQL query to retrieve content and metadata from columns in a database. Each row of data represents a document. Each document requires a unique identifier (a document reference) and content (body text). Optionally, it can have a title and other metadata.

Each database document store can have only one SQL query. A single SQL query can import one or all of your database documents into a document store, provided that the documents are all in a single database and that no authentication or authorization constraints require you to make multiple queries. For example, if you must specify more than one user name and password or more than one host, then you must construct more than one SQL query. You then require a database document store for each SQL query. Documents in separate databases require their own database document stores.

When constructing an import SQL statement, the following column names (or column aliases) have specific meaning. All are not case sensitive:

- DOC_REF – a unique token by which the document can be referred to for updates and deletes. A primary key column is most suitable for this.

  Sybase Search supports the following SQL types:

  - TINYINT
  - SMALLINT
  - INTEGER
  - BIGINT
  - REAL
  - FLOAT
  - DOUBLE
  - CHAR
  - VARCHAR – You can define the maximum length of VARCHAR.
- DOC_CONTENT – the text used as the body of the document. The text can be the content from TEXT or VARCHAR fields, or if used in conjunction with a content type value, it can be any of the supported MIME types.
- DOC_CONTENT_TYPE – the content type (or MIME type) of the document. When content is contained in the database in a binary format, DOC_CONTENT_TYPE provides the additional information required to decode it. For example, if the document content was UTF-8 encoded, plain text, then the content type is "text/plain; charset=UTF-8". Similarly, if the content field contains PDF bytes, the content type is "application/pdf". When the document content is binary and no content type is specified, Sybase Search attempts to decode it as plain text using the JRE default character set.

  If the content column contains a document binary and only the name or extension of the document is known (for example, *Report.pdf*), the name and extension can be supplied as the document content type. Sybase Search performs a MIME type for file extension lookup using its MIME Type map, and in these cases values, such as *C:\Documents\Report.pdf*, *Report.pdf*, *.pdf*, and *pdf* each represent the MIME type "application/pdf".
- DOC_LINK – a link to an external document on a file system. The link must be an absolute path, visible to Sybase Search. The document properties (where present) are extracted as metadata and Sybase Search uses the text as the document's body text.

Sybase Search treats all other column names and aliases as metadata and saves the information with the document as its metadata. If the metadata is to be indexed, its name and type must be known by the metadata manager. You are not required to supply metadata; however, as a best practice, you should supply a document TITLE. The document TITLE is shown on the document search results page.

Example of SQL query

The following example of a SQL query shows how a recruitment agency might import their current candidate CV resumés:

```
SELECT ID
      AS DOC_REF,                    /*INT*/
      PROFILE AS DOC_CONTENT,        /*VARCHAR*/
      CV AS DOC_CONTENT_2            /*BLOB*/
      CV_NAME AS DOC_CONTENT_TYPE_2,
      FIRST_NAME + ' ' + LAST_NAME AS TITLE,
      PREF_SALARY                    /*FLOAT*/
FROM
      CANDIDATES
WHERE
      LIVE=1
```

This example shows how the primary key column ID is used as a document reference, and how the document content (body text) is composed from both VARCHAR text in the database and a document on the file system. In this example, the MIME type of the document is not known; therefore, the original name of the document is passed to Sybase Search for it to query the appropriate MIME type. Also, a title is being constructed from the first and last name of the candidate. The preferred salary is saved as metadata.

Indexing document stores

Indexing is the process of collecting data about documents contained in a document store and storing its proprietary data structures, generically called indexes. After documents in a document store are indexed, they are available to be searched.

An indexing session describes all data collected during the pass of a document store's indexer. Data for all documents is collected during the first indexing session; subsequent indexing sessions collect data for new documents, modified documents, and deleted documents. Thus, the amount of data collected during two different indexing sessions can vary dramatically.

When creating a document store, you can request that Sybase Search immediately index the document store. You can also perform the following types of indexing after creating a document store:

- Incremental Index – click Incremental Index to rerun the indexing process over the saved document store configuration. All new documents are indexed; all updated documents are indexed again; and all deleted documents are removed from the indexes.
- Part Index – click Part Index to define specific directory paths containing documents that belong to a document store. When processing a Part Index, Sybase Search indexes only the documents found in the directory paths you define. If a document is already indexed, Sybase Search checks for modification and re-indexes, if necessary. If the document parameter is in the Sybase Search indexes but no longer exists on the file system, it is removed. The Part Index process saves time because it does not check the directory trees for new, modified, or deleted documents, which can save a significant amount of time for large document stores. Only the document parameters are considered.

  The Part Index process is primarily for use within OEM applications.

Note The document must exist within one of the document store's root directories. Documents not located in a valid root directory are ignored.

All data collected during an indexing session is stored in the indexing session's data buffer. The data buffer is a RAM-oriented data structure, where data is aggregated, ready to be written to an index stripe. This buffer is flushed when the maximum memory threshold has been exceeded (specified in the system property omniq.index.buffer.maxMemory). The buffer shares this memory allocation with the document store's active index stripe. See "Striping index data" on page 36.

Viewing indexed details

Sybase Search displays details of indexing activity for both the previous and any current indexing session with the details of the corresponding document store on the Document Stores page.

Table 3-3 summarizes the details of indexing activity displayed for each indexed document store.

Table 3-3: Indexing activity

Property	Value
Total	The total number of documents found
Indexable	The number of documents eligible for indexing
Selected	The number of documents selected for indexing
Skipped	The number of documents purposefully ignored
Deleted	The number of documents that have been indexed but no longer exist
New	The number of new unindexed documents found
Updated	The number of updated (changed since indexing) documents found
Unchanged	The number of indexed documents that have not changed
Failed	The number of documents that should have been indexed but were not, due to a problem

To view indexing data, click Index Information. Sybase Search displays the data on the Index Information page. Table 3-4 summarizes the data collected during an indexing session.

Table 3-4: Index information data

Property	Value
Documents Indexed	The total number of live and deleted documents in the indexes of all index stripes
Deleted Documents	The total number of documents in the indexes of all index stripes that reference deleted documents
Live Documents	The total number of documents in the indexes of all index stripes that reference live documents
Number of Stripes	The number of index stripes the indexed data is split across
Index Stripes	The details of each index stripe that the indexed data is split across

Striping index data

Index data is transferred from the data buffer and written to active or static stripes. Whether data is written to an active or a static index stripe is decided during the indexing session. The current active stripe stores all the data collected during the indexing session if it can accommodate it; otherwise, the active index stripe is emptied into a new static stripe, and all data collected during the indexing session is stored in the new static index stripe.

Active index stripes

Each document store's collection of index stripes contain exactly zero or one **active index stripe**. An active index stripe is a collection of RAM-oriented data structures—all of its data is stored in RAM while it keeps a copy on disk for persistence. An active index stripe is always writable, thus may contain data collected over numerous indexing sessions.

When an active index stripe is emptied into a static index stripe, these files are deleted and it is discarded. A new active stripe is created the next time an indexing session collects a sufficient amount of data to fit into an active index stripe.

Static index stripes

Each document store's collection of index stripes contain zero or more **static index stripes**. A static index stripe is a collection of disk-oriented data structures that you cannot change once they are written.

Viewing index stripe information

Each index stripe and details of its internal data structures are listed on the Index Information page. The details include the generic term, metadata indexes, and the data structures needed to track file system documents.

Table 3-5: Index stripe properties

Property	Value
Root	The location where the index stripe stores its data. The root property creates directories and data files in here as necessary.
Term Index Segments	The number of segments into which the term indexes are divided.
Metadata Index Segments	The number of segments into which the metadata indexes are divided.
Deleted Documents	The number of deleted documents for which this stripe still holds data (data which is purged on unification).
Live Documents	The number of live documents for which this stripe holds data.
Document lexicon	
Property	Value
Segments	The number of segments into which the document lexicon is divided.
Documents	The number of documents in the lexicon.
ID Range	The ID range of the document IDs (first to last).
Last Indexed	The name of the last document indexed and the time it was added.

Unifying an index stripe

Too many index stripes can eventually cause a bottleneck; therefore, you periodically should unify the stripes into a single stripe.

❖ **To unify an index stripe**

1 From the Document Stores page, select a document store and click Index Information. The Index Information page appears and displays the index stripe details.

2 Click Unify. The unification process runs. Sybase Search displays the progress of the unification process. Additionally, the unification process purges data marked for deletion and defragments the indexed data structures.

Managing document stores

Depending on the Sybase Search configuration, each satellite container provides a document store manager for each type of document store. For example, a file system document store manager contains file system document stores and lets you create file system document stores on the same container.

A document store manager manages zero or more document stores. Typically, there is one document store manager for each server where document indexing occurs. The document store manager for each document store that you create sets up document indexing on the different servers in the system.

As the administrator, you can import document stores and resize the query data cache.

Importing document stores

You can import document stores directly into a document store manager. You can also specify a document store manager at the time you create a document store. (See "Creating document stores" on page 28 for more information.)

❖ **To import a document store into a document store manager**

1 From the Document Store Manager page, select a document store manager.

2 Click Import. The Create Document Store page appears.

3 Follow the steps for creating a document store. See "Creating document stores" on page 28.

Resizing the query data cache

Query data cache allows more queries to be processed faster by caching commonly requested search and metadata terms in RAM. You can resize the maximum capacity of the query data cache to best meet the requirements of your environment.

❖ **To resize the query data cache**

1 From the Document Store Manager page, select a document store manager.

2 Click Resize. The Query Data Cache Capacity page appears.

3 Enter a value in megabytes that you want to increase or decrease the query data cache to. The value you enter must be at least 1MB.

4 Click Change. The query data cache is updated, and Sybase Search returns you to the Document Store Manager page.

Grouping document stores

You can group document stores into document groups. A **document group** lets users filter search results by the document stores defined in the selected document groups.

For example, a Sybase Search environment might include three document store managers on three separate machines; each document store manager has a "CV resumé" document store. You can create a "CV resumé" document group to include all three CV resumé document stores. You can then use the document group as a search parameter to indicate that search results should only come from the "CV resumé" document group.

Table 3-6 shows the properties of the Document Groups page.

Table 3-6: Document Groups properties

Property	Description
Name	The name of the document group
ID	The unique document group ID assigned to the group
Document store members	The document stores that are members of this group

Creating, editing, and removing document groups

You can create, edit, and remove document groups.

❖ **To create a document group**

1 From the Document Management page, select Document Groups. The Document Groups page appears.

2 From the Document Groups page, click Create New. The Create Document Group page appears.

3 In the Name field, enter a name used to uniquely identify the document group.

4 Select the document stores that you want to include in the group and click Add.

5 Click Create. You return to the Document Groups page and the new document group is added to the list.

❖ **To edit a document group**

1 From the Document Groups page, select the document group that you want to edit and click Edit. The Edit Document Group page appears.

2 Make the changes. You can change the name of the document group. You can also add or remove document stores from the document group.

❖ **To remove a document group**

1 From the Document Groups page, select the document group that you want to remove.

2 Click Remove. Sybase Search displays a message box to verify whether to remove the document group.

3 Click Yes. The document group is removed from the system.

Categorizing documents

Sybase Search lets you set up categories to help facilitate information retrieval. A **category** groups documents by content, independent of location or type of document store. You use categories to filter search results. You can also view lists of documents for each category. By setting up a well-organized category strategy, you can manage information by grouping documents of similar content.

Creating, editing, and removing categories

You set up categories by defining an initial query and a relevance threshold. You can also include metadata filtering. A category must have at least one search term or at least one metadata expression.

Using the initial query and the given threshold of the document's relevance percent, Sybase Search assigns a document to the category if its relevance percent is equal to or greater than the threshold.

For example, a query that consists of search terms and a minimum document relevance creates a category of documents that are grouped by their relevance to search terms defined in the given query. The use of the document relevance helps ensure that the documents in the category are valid matches.

Another example is a category query that consists of only metadata, such as "fileType = PDF". This creates a category that only contains PDF documents.

Users can search within a category about a certain subject, such as “England World Cup football.” Or, the user might simply use a category to filter search results, such as searching within a category of PDF documents.

By categorizing documents, you can create groups of documents on behalf of your users. Instead of searching for documents, a user could be presented with a pre-defined set of categories. The user can then browse the documents in each category.

❖ **To create a category**

1 From the Document Management page, select Categories. The Categories page appears.

2 Click Create. The Create Category page appears.

3 In the Name field, enter a text used to distinguish the category from others.

4 In the Description field, enter text to further describe the category.

5 Select the Query tab.

6 In the Search Terms field, enter a natural-language query. The more information you provide, the more accurate your results are. See “Searching across documents” on page 22 for more information.

7 In the Not Terms field, enter terms to indicate concepts dissimilar to those for which you are searching. See “Searching across documents” on page 22 for more information.

8 Click Document Groups.

9 From the group list, select one or more document groups to restrict your search.

10 Click Metadata. To include metadata in the category:

   1 Select a metadata parameter from the metadata list. You can add as many as five metadata parameters to the category. Click Add to add more metadata parameters.

   2 Select an operator. All metadata types support the equal to (=) operator. The integer and date types also support greater than or equal to(>=) and the less than or equal to (<=) operators.

   3 Enter a value for the metadata parameter. Table 3-1 on page 24 lists the predefined metadata parameters and types.

4 If the metadata parameter contains a value that consists of more than one term, select the Within Expression operator when the metadata parameter contains a value that consists of more than one term.

When you set the operator to AND, every term must be present in the document metadata for the match to succeed. When you set the operator to OR, only one of the terms must be present in the document metadata for the match to succeed.

5 If you have defined at least two metadata parameters, select the Across Expressions operator. When you set the operator to AND, both metadata parameters must succeed for the match to succeed. When you set the operator to OR, only one of the metadata parameters must succeed.

See "Searching across documents" on page 22 for more information.

11 Click Result Options. To set up result options:

1 From the Minimum Document Relevance list, select a percentage. The percentage you select defines the minimum relevance ranking that a document must score for it to be included within the search results. Documents with scores lower than the percentage that you enter are not returned.

2 Select the Score Unknown Terms check box to specify that terms unknown to the system – and, therefore, do not exist in any indexed document – are considered by the scoring algorithm.

12 Click Create. Sybase Search creates the category, assigns a unique system-generated numeric ID to it, and automatically adds documents that match the category criteria. Sybase Search displays the new category and list of relevant documents on the View Category page.

❖ **To edit a category**

1 From the Categories page, determine the category that you want to edit.

2 Click Edit. The Edit Category page appears.

3 Make the desired changes. See "To create a category" on page 42 for more information about each category property.

4 Click Update. Sybase Search updates the category properties and returns you to the Categories page.

❖ **To remove a category**

1 From the Categories page, determine the category that you want to remove.

2 Click View. The View Category page appears.

3 Click Remove. A message asking you to verify whether to remove the category appears.

4 Click OK. Sybase Search removes the category and returns you to the Categories page.

Viewing the contents of a document

You can view the contents of a document, open the document, and find documents that are similar to a selected document. You can view and open documents imported from a file system.

You only view and open source documents imported from a database if they have been imported as DOC_LINK references to file system documents. If a document is composed of multiple DOC_LINK file system documents, the first referenced document is returned for viewing.

❖ **To view the contents of a document**

1 From the View Category page, determine which document you want to view.

2 Click View Text to open the plain text of a document in a read-only browser.

3 Click View File to open the document in its native application.

4 Click Find Similar to display documents that contain similar content.

CHAPTER 4

Configuring Sybase Search

This chapter describes the key configuration parameters for containers, the hub, and modules. It includes tips on using the configuration files, and changing parameters.

Configuring the container XML file

Each container has an XML configuration file that determines if the container loads the hub and lists the modules to be loaded. You also use the configuration files to set system properties for the Java virtual machine (JVM) in which the container runs. The hub and modules run in containers, and thus share some configuration parameters.

The XML is formed with a root container tag enclosing zero or more System Property tags, exactly one Hub tag, zero or more Module tags, and zero or one Data tag.

The format is as follows:

```
<Container id="1" port="8001">
   <SystemProperty name="exampleName"
   value="exampleValue" />
   <Hub local="true" host="127.0.0.1" port="7000"
   bindName="Hub" logEvents="true" />
   <Module id="101" class="com.omniq.xmp.ExampleModule"
   name="Example Module" />
   <Data directory="G:\example\data" />
</Container>
```

The modules can contain zero or more HttpHandler tags, which in turn can contain zero or more Param tags. For example:

```
<Module id="101" class="com.omniq.xmp.ExampleModule"
   name="Example Module">
   <HttpHandler class="com.omniq.xmp.ExampleHandler"
   resourceURI="/handler/example">
       <Param name="exampleName" value="exampleValue"/>
   </HttpHandler>
</Module>
```

The hub

The hub is the core component of the system. It is a special module that is the global coordinator of Sybase Search. The container that loads the hub container also runs a Java RMI registry to listen for remote requests. Satellite containers load a hub facade to handle communication with the real hub. All queries and administration requests are negotiated by the hub.

Configuration and ID conventions

Sybase Search example configuration files can be obtained upon request, and the quickest way to obtain configuration files is to install the required container.

The files for a single-server configuration are located in *<installLocation>\OmniQ\config\Container.1.xml*.

Multiple-server configuration requires more than one file. One configuration file is for the hub container and one configuration file is required for each satellite container.

The files for multiple-server configuration are respectively located in:

- *<installLocation>\OmniQ\config\Container.1.xml*
- *<installLocation>\OmniQ\config\Container.2.xml*

Containers, hub facades, and modules are not automatically assigned unique IDs (UIDs)—you must configure them manually. The UIDs must be within the range of 1 to the UID Generator's seed value, which is 1,000 by default. See "Setting Unique ID (UID) Generator parameters" on page 51.

If a container or module is assigned an ID greater than the seed value, it may conflict with an internally generated ID and cause an unexpected error later.

Because these UIDs are split across several files, you must employ a numbering convention. The example two-server configuration files use the following conventions:

- Container ID – a value from 1– 99.
- Container XML – includes the container ID in its name, for example, `Container.1.xml`.
- HTTP listener – the container's HTTP listener binds to the port number 8000 plus the container's ID. For example, the port is 8001 for Container 1 and 8002 for Container 2.
- Hub container – always binds the RMI Registry on port 7000.
- Hub facade ID – on satellite containers this is 10 times the container ID. For example, the hub facade ID for Container 2 is 20.

  Note An exception is that the default Web application always allocates its hub facade ID as 999 as it does not need to follow the other conventions.

- Modules – each module has the ID of 100 times the Container ID + *N*. For example, the first module ID on Container 1 is 101, the second is 102, the third is 103 and so on.

The single server configuration does not strictly follow this convention as all IDs are visible in the same file, making the assigning of duplicate IDs less likely. If the Sybase Search installation requires more than nine servers, a different convention is necessary.

Table 4-1 shows the attributes for the container tag.

Table 4-1: Container tag

Attribute	Default value	Description
id	None	The unique ID of the container. This value is used to identify the container when it registers itself with the hub.
port	None	The TCP/IP port on which the container's embedded HTTP server listens.

Table 4-2 shows the attributes for the SystemProperty tag. The system properties include JVM settings, Stellent SearchML settings, and global indexing and querying parameters for modules loaded within the container.

Table 4-2: SystemProperty tag

Attribute	Default value	Description
name	None	The name of the Java system property to set. In other words, the name you use within the Java process when using the java.lang.System.getProperty (java.lang.String) method.
value	None	The string value to associate with the property name.

Table 4-3 shows the attributes for the hub tag.

Table 4-3: Hub tag

Attribute	Default value	Description
local	false	When set to true, the real hub is loaded into the current container. Otherwise, the container loads a hub facade.
id	None	This is the unique ID of the hub facade, which is used when the hub facade registers itself with the real hub. If the hub is local, this attribute is not required.
host	127.0.0.1	If the hub is not local, the hub facade uses this value to contact the real hub on the RMI registry.

Attribute	Default value	Description
port	None	The TCP/IP port on which the RMI registry started by the hub container is bound. When the hub is local, the port is used when starting the RMI registry. When the hub is not local, the port is used to connect to the RMI registry to access the real hub.
bindName	Hub	The name by which the hub is bound on the RMI registry. When the hub is local, bindName is used to bind the hub. When the hub is not local, bindName is used to look up the hub.
logEvents	false	This value indicates whether or not the event log should be enabled. The location of the hub is irrelevant.
logDirectory	*<data.directory>\log*	

Table 4-4 shows the attributes for the module tag.

Table 4-4: Module tag

Attribute	Default value	Description
id	None	The unique ID of the module, used to identify the module when it is registered with the hub.
name	None	The name of the module.
class	None	The name of the Java class that is the module.
enabled	true	If set to false, the module is not loaded.

Table 4-5 shows the attributes for the HttpHandler tag.

Table 4-5: HttpHandler tag

Attribute	Default value	Description
class	None	The name of the Java class that is the HTTP handler (the resource).
resourceURI	None	The HTTP URI of the HTTP handler resource. This is used to complete the URL, for example, http://<*container.host*>:<*container.port*>/<*resourceURI*>.
name	None	The name of the parameter to pass to the HTTP handler.
value	None	The string value to associate with the parameter name.

Configuring modules

This section describes the Sybase Search modules and their configurations. Each module runs in a container and, with a few restrictions, can either be run in its own separate container on different servers, or grouped with other modules within a single container.

The available modules are:

- Setting Unique ID (UID) Generator parameters
- Setting Document Group Manager parameters
- Setting Text Manager parameters
- Setting Term Lexicon Manager parameters
- Setting Term Lexicon Manager Delegate parameters

- Setting Metadata Manager parameters
- Setting Metadata Manager Delegate parameters
- Setting Query Manager parameters
- Setting Repository Manager parameters
- Setting Filter Factory parameters
- Setting Category Manager parameters
- Setting Database Document Store Manager parameters
- Setting File System Document Store Manager parameters

Setting Unique ID (UID) Generator parameters

The Unique ID Generator settings are loaded through the *UIDGeneratorModule.default.xml* configuration file. Table 4-6 shows the parameters in this file.

Table 4-6: UIDGeneratorModule.default.xml parameters

Parameter	Default	Description
filename	uid.dat	The file that stores the next unique ID.
alwaysOpen	false	If set to true, the underlying Java class leaves the file handle open to the file name above.
seed	1,000	The UID Generator seed starts from 1,000, as numbers less than 1,000 are reserved by Sybase Search as module IDs.

Setting Document Group Manager parameters

The Document Group Manager module is contained in the hub container and does not have a configuration file associated with it, because initially the system contains no predefined document groups.

Setting Text Manager parameters

The Text Manager module settings are loaded through the *TextModule.default.xml* configuration file. Table 4-7 shows the parameters in this file.

Table 4-7: TextModule.default.xml parameters

Parameter	Default	Description
min.term.length	2	The minimum term length deemed valid for indexing. This is not taken into account in the list of preserved terms and does not apply to single-digit terms.
max.term.length	20	The maximum term length deemed valid for indexing. This value must match the Term Lexicon Manager parameter term.length.max.
stopwords.filename	*stopwords_en.txt*	Contains a list of stopwords to remove during the indexing and querying processes to improve system performance. See "Defining the list of stopwords" on page 62.
preserved.terms.filename	*preserved_terms_en.txt*	Contains the list of preserved terms that are not stemmed during indexing. The list can also include terms less than the minimum term length defined in the min.term.length parameter. See "Defining the list of preserved terms" on page 62.
term.splitter.class	*com.isdduk.text.BreakIteratorSplitter*	Specifies the Java class used for breaking text into separate words. The default BreakIteratorSplitter handles all double-byte character sets.
term.stemmer.class	*com.isdduk.text.Porter2Stemmer*	Specifies the Java class used for term stemming. The default Porter2Stemmer is for English text.
query.augmentor.filename	*query_aug_en.txt*	Contains a list of synonyms and acronyms. See "Augmenting queries" on page 63.
parsers.filename	*Parsers.xml*	The name of the file in the *config* folder that contains the list of text parsers.

You can set the term splitter and stemmer classes to language-independent classes or to language-specific classes. Language-specific stemmers allow an increase in system performance when Sybase Search is going to index documents in one language only.

Setting Term Lexicon Manager parameters

The Term Lexicon Manager settings are loaded through the *TermLexiconModule.default.xml* configuration file. Table 4-8 shows the parameters for the Term Lexicon Manager.

Table 4-8: Term Lexicon Manager parameters

Parameter	Default	Description
term.length.max	20	The maximum term length deemed valid for indexing. This value must match the Text Manager parameter max.term.length. See "Setting Text Manager parameters" on page 51.
cache.capacity	131,072	The number of terms stored in memory to improve indexing and querying performance.
cache.useRootChildrenCache	true	If set to true, the underlying term lexicon data structures cache some of their structure in memory to improve indexing and querying performance.
unify.size.threshold	10,000	Determines how many terms in each term lexicon segment are stored in memory before being written to disk.
unify.idle.threshold	120,000	The time, in milliseconds, that the Term Lexicon Manager remains idle before unifying the pending terms. The idle time is restarted when a new term is added, or when an existing term is looked up.
number.of.segments	20	The number of term lexicon segments. For maximum efficiency, the value should be equal to the term.length.max.
minimization.factor	50	The branching factor of the underlying term lexicon segments. This parameter affects the lookup performance of the Term Lexicon Manager. Do not change this value without consulting Sybase Technical Support.

Setting Term Lexicon Manager Delegate parameters

The Term Lexicon Manager Delegate allows terms and their unique IDs to be cached locally. This saves the remote module from excessive communication with the hub's Term Lexicon Manager. The Term Lexicon Manager Delegate settings are loaded through the *TermLexiconModuleDelegate.default.xml* configuration file.

The only parameter in the *config* file is cache.capacity, which represents the number of terms that can be cached locally.

Setting Metadata Manager parameters

The Metadata Manager settings are loaded through the *MetadataModule.default.xml* configuration file. Table 4-9 shows the parameters in this file.

Table 4-9: MetadataModule.default.xml parameters

Parameter	Default	Description
metadata.filename	*Metadata.ser.gz*	The name of the file to where the metadata fields are serialized. Do not change this parameter.
uid.filename	uid.dat	The name of the file that stores the next unique ID, which is used when creating new metadata fields. Do not change this parameter.

Setting Metadata Manager Delegate parameters

The Metadata Manager Delegate allows metadata fields and their unique IDs to be cached locally. This saves the remote module from excessive communication with the hub's Metadata Manager. The Metadata Manager Delegate settings are loaded through the *MetadataModuleDelegate.default.xml* configuration file.

Table 4-10: Metadata Manager Delegate parameter

parameter	Default	Description
metadata.filename	*Metadata.ser.gz*	The name of the file to where the metadata fields are serialized to. Do not change this parameter.

Setting Query Manager parameters

The Query Manager settings are loaded through the *QueryModule.default.xml* configuration file. Table 4-11 shows the Query Manager parameters.

Table 4-11: Query Manager parameters

parameter	Default	Description
cache.termStats.capacity	131,072	The number of term statistics stored in memory to improve querying performance.
queryRunnerPool.size	20	The number of concurrent threads used to run queries.
queryParsers.filename	*QueryParsers.xml*	The name of the file in the *config* folder that contains the list of query parsers.

Setting Repository Manager parameters

The Repository Manager has no configuration settings and is used to allow other containers to pass on the text from documents located in other containers.

Setting Filter Factory parameters

The Filter Factory parameters are loaded through the *FilterFactory.default.xml* configuration file.

The list of default filters in the configuration file are:

- Default HTML filter
- Default EML filter
- SearchML export multi-filter
- SearchML filter

Each filter specifies a number of settings, which determine which class is loaded for the filter, which paragraph extractor is used, and the MIME types to which the filter applies. Table 4-12 shows the filter setting parameters.

Table 4-12: Filter settings

Parameter	Default	Description
className	None	The Java class that defines the filter.
extractorClassName	None	The Java class used for extracting paragraphs from the filtered text.
mimeTypes	None	The list of MIME types that are associated with the filter.
timeout	45,000	Indicates the time in milliseconds the filter waits while filtering a document. If the filter exceeds the given time, the filter aborts. This parameter is used mainly by the Stellent filter.
keepTempFiles	false	If set to true, the filter keeps any temporary files produced during the filtering process. This is used mainly by the Stellent filter.

In addition to the filter-specific settings, there are a number of general filter settings that help the extractors determine the paragraphs. The filter ensures that each paragraph is between the minimum and maximum lengths and aims for the ideal paragraph length.

Table 4-13 shows the paragraph length settings.

Table 4-13: Paragraph settings

Parameter	Default	Description
default.minParaLen	250	The minimum number of characters in a paragraph
default.idealParaLen	500	The ideal number of characters in a paragraph
default.maxParaLen	1,000	The maximum number of characters in a paragraph

Setting the Default HTML filter parameters

The default HTML filter is a custom filter used to parse HTML files.

Table 4-14 shows a list of the parameter settings used by the default HTML filter.

Table 4-14: Default HTML filter parameters

Parameter	Value
className	com.omniq.filter.html.HTMLFilter
extractorClassName	com.omniq.filter.StandardExtractor
mimeTypes	text/html
timeout	N/A
keepTempFiles	N/A

Setting the Default EML filter parameters

The default EML filter is a custom filter used to parse EML files that are used by some e-mail systems.

Table 4-15 shows the parameter settings used by the default EML filter.

Table 4-15: Default EML filter parameters

Parameter	Value
className	com.omniq.filter.eml.EMLFilter
extractorClassName	com.omniq.filter.eml.StandardExtractor
mimeTypes	message/rfc822
timeout	N/A
keepTempFiles	N/A

Setting the SearchML export multi-filter parameters

The SearchML export multi-filter is the default filter used to parse all other MIME types that do not have their own specific filter. The output from the SearchML export multi-filter is an XML document that contains the raw text and associated document metadata.

Table 4-16 shows a list of the parameter settings used by the SearchML export multi-filter.

Table 4-16: SearchML multi-filter parameters

Parameter	Value
className	com.omniq.filter.stellent.SearchMLFilterExport
extractorClassName	com.omniq.filter.StandardExtractor
mimeTypes	*
timeout	45000
keepTempFiles	false

Setting the SearchML filter

The SearchML filter is an internal filter used to parse the XML output from the SearchML export multi-filter as given above.

Table 4-17 shows a list of the parameter settings used by the default SearchML filter.

Table 4-17: SearchML parameters

Parameter	Value
className	com.omniq.filter.stellent.SearchMLFilter
extractorClassName	com.omniq.filter.StandardExtractor
mimeTypes	text/x-searchml
timeout	N/A
keepTempFiles	N/A

Setting Category Manager parameters

The Category Manager settings are loaded through the *CategoryModule.default.xml* configuration file. Table 4-18 shows the parameters in this file.

Table 4-18: Category Manager parameters

Parameter	Default	Description
categoryRunnerPool.size	20	The number of concurrent threads used to run category queries
queryParsers.filename	QueryParsers.xml	The name of the file in the config folder that contains the list of query parsers

Setting Database Document Store Manager parameters

The Database Document Store Manager settings are loaded through the *DBDocumentStoreModule.default.xml* configuration file. Table 4-19 shows the parameters in this file.

Table 4-19: Database Import Manager parameters

Parameter	Default	Description
cache.queryData.capacityInBytes	52428800	The maximum amount of memory allowed for Document Store Manager query data cache

Setting File System Document Store Manager parameters

The File System Document Store Manager settings are loaded through the *FSDocumentStoreModule.default.xml* configuration file. Table 4-20 shows the parameters in this file.

Table 4-20: File System Import Manager parameters

Parameter	Default	Description
cache.queryData.capacityInBytes	52428800	The maximum amount of memory allowed for the Document Store Manager's query data cache

Optimizing Sybase Search

You can optimize Sybase Search if you know that all the source documents are in one language. This optimization includes using stemming algorithms, stopwords, and preserved terms.

Configure these settings using the Text Manager and Query Manager.

Processing metadata values

Parsers are used for processing metadata values, which are generally received as string key/value pairs. While document body text is processed by the system term splitter and stemmer, metadata often must be handled differently (as metadata values can be not only strings, but also numeric and date types). The parsers loaded by the Text Manager are referenced in the metadata field parser and query parser XML configuration files.

There are four types of parsers:

- String
- Numeric decimal
- Numeric integer
- Date (time)

A string parser is always handled by internal classes. You can build custom numeric and dates parsers and plug them into the system if necessary.

Table 4-21 shows the attributes for the Parser tag.

Table 4-21: The Parser tag

Attribute	Default	Description
identifier	None	The Parser instance's identifier. This must be a name and a unique ID separated by an underscore (_).
class	None	The Java implementation class.

Table 4-22 shows the attributes for the Param tag.

Table 4-22: The Param tag

Attribute	Default	Description
name	None	The name of the parameter to pass to the parser.
value	None	The string value to associate with the parameter name.

Sybase Search comes with the preconfigured parsers, shown in Table 4-23, which are adequate for most common metadata types.

Table 4-23: Preconfigured parsers

Name	float_1
Class	**com.isdduk.text.SimpleFloatParser** This class parses strings representing decimal numbers into actual decimal numbers. For example, the string "3.142" is parsed into Java float 3.142.
Name	integer_2
Class	**com.isdduk.text.IntegerParser** This class parses strings representing an integer number into an actual integer number; any floating-point information is discarded. For example, both "3" and "3.142" are parsed into Java int 3.
Name	dateUK_3
Class	com.isdduk.text.DateFormatParser
Name	dateMs1970_4
Class	com.isdduk.text.Ms1970DateParser
Parameter	Name – roundTo. Value – choose a year, month, day, hour, minute, second, or any other value to denote no rounding should take place. This class is date parser, which effectively parses strings representing long integer (64-bit) numbers, which themselves represent dates as the number of milliseconds since 1 January 1970. The preconfigured instance rounds dates to the nearest day (UTC).
Name	intB2KB_5

Class	**com.isdduk.text.B2KBIntParser** This class parses strings representing byte-size numbers and converts them into kilobyte-size numbers. For instance, the string “2048” (bytes) is parsed as Java int 2 (kilobytes).
Name	datePDF_6
Class	**com.isdduk.text.PDFDateParser**
Parameter	Name – roundTo. Value – choose a year, month, day, hour, minute, second, or any other value to denote no rounding should take place. This class handles the PDF date format, in which dates are formatted “D:20030602143803+01'00'”. The preconfigured instance rounds dates to the nearest day (UTC).

Configuring parsers to query metadata fields

The settings of query parsers are loaded through the *config* file as specified in the queryParsers.filename parameter in the Query Manager configuration file.

This enables the Sybase Search administrator to configure parsers for querying metadata fields. In some cases, you might want to search metadata fields using formats different from those that were indexed. For instance, a date metadata field might be indexed in the YYYY-MM-DD format yet searched on using a DD/MM/YYYY format. Table 4-24 shows the metadata field tag attributes.

Table 4-24: Metadata field tag attributes

Attribute	Default value	Description
name	None	The internal name of the metadata field to which this setting applies.
parser	None	The identifier of the parser used to parse the query metadata values.

Note You can change the query parser configuration at any time; however, changes take effect only after you restart the container hosting the Query Module.

Defining the list of stopwords

Stopwords are common words such as “I,” “a,” “an,” “the,” and so on, that are ignored during the indexing or querying process. Removing the most common words during the indexing process keeps index sizes smaller, which enhances performance.

You can change the list of stopwords in one of two ways:

- Edit the list of words in the default stopwords file located in *<installLocation>\OmniQ\config\stopwords_en.txt*, or
- Create a new stopwords file and configure the Text Manager to read from the new file, by editing *<installLocation>\OmniQ\config\TextModule.default.xml* and changing the value of the stopwords.filename parameter to point to the new file.

Note The stopword list must be UTF-8 encoded. Because the words on the stoplist are ignored when you index documents (in other words, the document is indexed as if the words on the stoplist did not exist), you must make any changes to the stoplist before you index. If you have already indexed your documents, and you add new stopwords, the words are not included in your query, but the disk space consumed by that word’s associated data is not reclaimed until you index your documents again.

Removing stopwords after you have already indexed your documents has no effect until you index your documents again.

Defining the list of preserved terms

You can use preserved terms to ensure that some terms are *not* removed as part of the indexing and querying processes. For example, the term “US” would be removed from any extracted text if the term “us” was entered in the list of stopwords. The case-sensitive list of preserved terms ensures that “us” will be removed, but “US” is indexed and made available to the query calculations.

You can change the list of preserved terms in one of two ways:

- Edit the list of words in the default preserved terms file, *preserved_terms_en.tx*, located in *<installLocation>\OmniQ\config*, or

- Create a new preserved terms file and configure the Text Manager to read from the new file by editing *<installLocation>\OmniQ\config\TextModule.default.xml* and changing the value of the preserved.terms.filename parameter to point to the new file.

Note The preserved term list must be UTF-8 encoded and changed before you index any documents, as preserved terms require special handling during indexing. If you have already indexed documents, changing the preserved terms has no effect, because the terms must still be queried exactly as before to produce matches (as the terms are fixed in the indexes).

Augmenting queries

In Sybase Search, the use of synonyms and acronyms is collectively called **query augmentation**. Synonyms are implemented as lists of words that are considered to have the same meaning. For example:

- drowsy, lethargic, listless, sleepy
- holiday, vacation

When a term featured in a list is used as a query parameter, all the other words in the list are appended to the query. For example, the query `The medicine made me drowsy`, when augmented using the synonym examples above, becomes `The medicine made me drowsy, lethargic, listless, sleepy`.

Acronyms are implemented as a list of keys (or a single key) with a corresponding list of values. In the following example the keys “HTML” and “HTM” have the values “Hypertext Markup Language”:

- HTML, HTM = Hypertext Markup Language
- USA, US = United States of America

Acronyms can augment a query in two ways:

- Acronym expansion – when a term featured in an acronym key list is found in a user’s search terms, all the corresponding values are added to the original query. For example, the query `How to write HTML documents`, when augmented with the acronym examples above, becomes `How to write HTML Hypertext Markup Language documents`.

- Acronym resolution – when a list of terms featured as an acronym values list is found in a user's search terms, all the corresponding keys are added to the original query. For example, the query `How to write Hypertext Markup Language documents`, when augmented with the acronym examples above, becomes `How to write Hypertext Markup Language HTML HTM documents`.

You can change the list of synonyms and acronyms in one of two ways:

- Edit the list of words in the default synonyms and acronyms file located in *<installLocation>\OmniQ\config\query_aug_en.txt*, or
- Create a new query augmentation file and configure the Text Manager to read from the new file, by editing *<installLocation>\OmniQ\config\TextModule.default.xml* and changing the value of the query.augmentor.filename parameter to point to the new file.

Note The query augmentation list must be UTF-8 encoded. Synonyms and acronyms are processed at runtime, so you can edit the augmentation list without having to index any documents again. However, you must restart Sybase Search to load the new synonym and acronym lists.

Configuring metadata fields

Sybase Search supports four types of metadata fields:

- TEXT
- DATE
- FLOAT
- INT

Each of these types supports a number of different parsers, which format or modify the metadata in different ways. (See "Processing metadata values" on page 59.) For example, different DATE parsers parse the date value differently depending on the date format specified.

Each metadata field is configured to be one of these four types and use one of the valid parsers.

Sybase Search cannot automatically extract new metadata fields from documents, because the metadata type and parser must be specified by the Sybase Search administrator in advance.

Setting TEXT metadata fields

TEXT metadata corresponds to any metadata field that contains words or characters. If any date or numeric information is treated as text, then each digit or number is treated simply as another character.

You can set the TEXT metadata parser to one of the internal text parsers that are set by the Text Manager module. These are set as either TEXT_STANDARD or TEXT_FILENAME.

TEXT metadata fields do not support range searching.

Setting DATE metadata fields

DATE metadata corresponds to any metadata field that can be parsed into a date. The exact parsing of the date depends on the date parser's settings. For example, the parser may have the format DD/MM/YYYY or YY-MM-DD.

DATE metadata fields support range searching.

Setting FLOAT metadata fields

FLOAT metadata corresponds to any metadata field that can be parsed into a numeric value. The exact parsing of the numeric value depends on the parser's settings.

FLOAT metadata fields support range searching.

Setting INT metadata fields

INT metadata corresponds to any metadata field that can be parsed into an integer value. The exact parsing of the numeric value depends on the parser's settings. For example, the com.isdduk.text.IntegerParser parser simply attempts to convert the metadata to an integer, while the com.isdduk.text.B2KBIntParser parser attempts to convert the metadata to an integer and then divide the result by 1024 to get the value expressed in terms of kilobytes (KB) instead of bytes (B).

INT metadata fields support range searching.

Defining metadata fields

The list of metadata fields is loaded through the configuration file as specified in the metadata.filename parameter (*Metadata.xml*) in the Metadata Manager configuration file. Each metadata field in *Metadata.xml* is specified by the parameters shown in Table 4-25.

Table 4-25: Metadata field parameters

parameter	Default	Description
name	None	The internal name of the metadata field; one word with no spaces. This name is used in XML queries over HTTP.
displayName	None	The human-readable name for the metadata field.
type	None	The metadata field type—can be one of TEXT, DATE, FLOAT and INT.
parser	None	The parser used to format the metadata field into the format Sybase Search will use. The parser must be defined in the *Parsers.xml* file.
indexable	false	If set to true, Sybase Search indexes any document data found for this metadata field.

Adding new metadata fields

When source documents contain metadata fields that are not listed in the default set, you can add them:

❖ **To add metadata fields**

1 Using a text editor, open the *Metadata.xml* located in *<installLocation>\OmniQ\config*.

2 Anywhere within the XML Metadata tag, add a new field tag specifying the following attributes:

- Name – the name of the metadata field inside the document.
- DisplayName – specifies the way the metadata field displays on the search page.
- Type – specifies whether the metadata type is TEXT, DATE, FLOAT, or INT.
- Parser – the name of the parser used to parse the metadata field for indexing. For TEXT, use one of the internal parsers (TEXT_STANDARD or TEXT_FILENAME). Otherwise, use a parser listed in the *Parsers.xml* file).
- Indexable – set this property to true to index this metadata field.

3 Save and close the file.

For example, if the new metadata field is Customer ID, the new field tag would look similar to this:

```
<Field name="custId" displayName="Customer ID" type="INT"
parser="integer_2" indexable= "true" />
```

Note You must add the new metadata field before indexing any documents.

If the metadata field requires parsing from a nonstandard string (for example, to change the customer ID portion of the string "CUST-98334" to an INT), refer to "Developing and configuring customized parsers" on page 84.

If you need to search the new metadata field in a different format from that in which it was indexed, you must configure a new query parser. For example, if INT metadata field values have been parsed from strings such as "CUST-98334," but you do not want to type the "CUST-" prefix for searching, you can add a new query parser configuration as follows:

1 Using a text editor, open *QueryParsers.xml*, which is located in *<installLocation>\OmniQ\config*.

2 Anywhere within the XML QueryParsers tag, add a new MetadataField tag, specifying the following attributes:

- Name – the internal name of the metadata.
- Parser – the name of any parser listed in the *Parsers.xml* configuration file.

3 Save and close the file.

4 Restart Sybase Search.

Using the above example, the new MetadataField tag might look similar to this:

```
<MetadataField name="custId" parser="integer_2" />
```

Note You can change the query parser configuration at any time.

Configuring MIME types

The list of MIME types that Sybase Search can index is in the *MimeTypeMap.default.xml* configuration file. In the configuration file, each MIME type is assigned a type and is marked as to whether or not it is supported.

A MIME type might have several extensions, each of which may or may not be indexable. The list of MIME types allows Sybase Search to index only those document types that may contain valid text data. Common formats such as plain text, Microsoft Office documents, Adobe PDF documents, and HTML files are indexable by default, whereas executable MIME types are not.

You can add custom MIME types and the appropriate text filter in the *FilterFactory.default.xml* file.

Configuring modules using system parameters

Many shared module settings are configured using SystemProperty tags in the container XML configuration file. These properties are set as JVM system properties and are accessible to all classes loaded in the container. Properties set in this manner are “container-global.”

You can enter numeric parameters using several formats:

- Plain integers – for example, 20
- K – for example, 20K = 20 x 1000
- M – for example, 20M = 20 x 1000K

- KB – for example, 20KB = 20 x 1024 bytes
- MB – for example, 20MB = 20 x 1024K
- GB – for example, 20GB = 20 x 1024MB

These formats allow high values to be entered as parameters while keeping the parameters easy to read. They also avoid the "missing zero" problem that can sometimes occur when entering parameters that have a high number of trailing zeros.

Indexing processes

Sybase Search stores its data in a number of proprietary data structures, generically called indexes. See "Indexing document stores" on page 34 for more information.

Indexing involves three different processes. The first two are fundamental indexer processes and might occur numerous times during one indexing session. The third process occurs as a maintenance operation. These processes are:

- Filtering, or parsing, documents and extracting data in memory
- Writing processed data to index stripes on disk
- Unifying index stripes on disk

Extracting data into memory

The first indexing process has a threshold for restricting the amount of memory the extracted data buffer can consume before the data is written to disk (process 2). The greater the memory allocation, the more efficient is the entire indexing process, because more data can be handled at once.

Parameters that affect the extraction process are shown in Table 4-26.

Table 4-26: General upload parameters

Parameter	Default	Description
omniq.index.buffer.maxMemory	10MB	The indexing process is more efficient if many documents are indexed in a batch. The buffer's maximum memory allocation determines how many documents are processed in each batch.

Parameter	Default	Description
omniq.indexer.maxDocumentSize	10MB	Sets the maximum document size to be indexed by Sybase Search. **Note** Very long documents have an adverse effect on the query results.

Writing data to disk

The second process is when the buffered data is written to the indexes. There are two main sets of parameters that affect this stage—the rate at which the data is written to the indexes (to reduce CPU and disk contention), and the index settings themselves.

Parameters that affect the write process are shown in Table 4-27.

Table 4-27: Index parameters

Parameter	Default	Description
omniq.indexer.sleepDurationMillis	20	The time, in milliseconds, the indexer thread sleeps during indexing to allow other CPU-intensive applications to run.
omniq.indexer.sleepFrequency	20	Indicates the number of omniq.indexer.sleepFrequency cycles the indexer thread will sleep.
omniq.index.term.numSegments	5	The number of segments helps to distribute the indexed data across a number of files, reducing the “seek” times of large files.
omniq.index.term.minimizationFactor	20	The branching factor of each index segment. This parameter affects the lookup performance of the index segment.
omniq.index.term.useRootChildrenCache	true	If set to true, the index segments cache some of their structure in memory to improve indexing and querying performance.
omniq.index.metadata.numSegments	2	The number of segments helps to distribute the indexed data across a number of files, reducing the seek times of large files.
omniq.index.metadata.minimizationFactor	10	The branching factor of each metadata index segment. This parameter affects the lookup performance of the metadata index segment.
omniq.index.metadata.useRootChildrenCache	true	If set to true, the metadata index segments cache some of their structure in memory to improve indexing and querying performance.

Parameter	Default	Description
omniq.lexicon.document.maxKeyLength	256	The maximum document file path length deemed valid for indexing.
omniq.lexicon.document.minimizationFactor	20	The branching factor of each document lexicon segment. This parameter affects the lookup performance of the document lexicon segment and should not be changed without consulting with technical support.
omniq.lexicon.document.useRootChildrenCache	true	If set to true, the document lexicon segments will cache some of their structure in RAM to improve indexing and querying performance.
omniq.lexicon.reverseDocument.numSegments	4	The number of segments helps to distribute the indexed data across a number of files, reducing the seek times of large files.

Unifying index stripes

The unifying process is for maintenance and optimization. This can take place only after a document store has been indexed again, which in turn produces new Index Stripes. See“Unifying an index stripe” on page 38 for more information.

Parameters which affect the unifying process are shown in Table 4-28.

Table 4-28: Unifying parameters

Parameter	Default	Description
omniq.unifier.sleepDurationMillis	20	The time, in milliseconds, that the unifier thread sleeps during unifying to allow other CPU-intensive applications to run.
omniq.unifier.sleepFrequency	100	Indicates the number of omniq.unifier.sleepFrequency cycles the unifier thread will sleep.
omniq.unifier.termMapSizeSoftLimit	40K	The limit of the number of terms processed in each unifying batch.
omniq.unifier.termMapSizeInBytesSoftLimit	32MB	The memory limit used for processing the terms in each unifying batch.
omniq.unifier.metadataMapSizeSoftLimit	40K	The limit to the number of metadata processed in each unifying batch.
omniq.unifier.metadataMapSizeInBytesSoftLimit	32MB	The memory limit used for processing the metadata in each unifying batch.

Warning! The index and lexicon parameters are critical to how the system performs—do not modify them without consulting with a Sybase Sybase Search support engineer.

Setting Query parameters

All queries run against Sybase Search are affected by the query parameters. The document scores can be scaled up or down using the confidence parameter, and the linking parameters affect all "find similar" queries.

Table 4-29 shows the query parameters.

Table 4-29: Query parameters

Parameter	Default	Description
omniq.query.termLimit	30	The maximum number of terms in a query. If the user's query exceeds this number, Sybase Search selects the most important omniq.query.termLimit number of terms from the query to use as the internal query.
omniq.query.confidence	125	Sybase Search generates its own scaling factor when converting internal document relevance scores to a more user-friendly percentage score. This scaling can be influenced by the omniq.query.confidence and has the effect that a higher confidence lowers the overall scores, while a lower confidence raises the overall scores.
omniq.query.linking.default.minDocRel	5	The minimum document relevance for a linking query can be specified on a per-query basis, but this value is used when the minimum document relevance is not specified.
omniq.query.linking.minTerms	5	The minimum number of terms that are generated automatically by Sybase Search to be used as a linking query.
omniq.query.linking.maxTerms	10	The maximum number of terms that are generated automatically by Sybase Search to be used as a linking query.
omniq.query.linking.confidence	50	The omniq.query.linking.confidence parameter works in the same way as omniq.query.confidence does, except for linking queries instead of normal queries. Linking queries tend to generate lower document scores, as the generated linking query can cover many different topics. To compensate, the confidence parameter is low to raise the overall linking query scores.

Setting up metadata paragraph files

The metadata paragraph files (MPFs) are where Sybase Search stores the metadata and text of the files it has indexed. This data is used in constructing result sets and for generating plain-text versions of the indexed documents.

The MPF is a custom data structure—each contains the metadata and body text of a number of indexed documents (“Configuring MPFs” on page 73) in a compressed format. The first group of MPFs is created in the 0 (zero) directory, and subsequent groups are numbered sequentially beginning with 1.

Configuring MPFs

The MPF classes utilize a strategy to best compress all the paragraphs from documents, favoring documents of average length (where the average length is implied from the MPF configuration). Each paragraph is written to disk in one of two ways:

- The paragraph is entered into a paragraph group that is compressed as a whole and written to disk.
- The paragraph is compressed and written to disk individually.

The first technique is employed initially, as the compression scheme works better with more data—thus the paragraphs take up less space on disk. The second technique is employed when the paragraph group allocation is exhausted.

The paragraphs are not all written together, as it is often necessary to read individual paragraphs from disk (and compressing all the paragraphs together forces the application to read and decompress all paragraphs to access the sole paragraph required). The grouping provides a balance between data compression and disk I/O.

The number of paragraphs in any one paragraph group is not fixed; groups accept new paragraphs until the data buffer’s soft limit is reached. “Soft” indicates that a limit can be exceeded, but the group is then closed. The ideal scenario is when all the paragraphs from a document fit exactly within the allocated number of paragraph groups. Unused paragraph groups result in redundancy.

You can configure the paragraph grouping using the MPF parameters shown in Table 4-30. The MPF parameters are defined for all document stores in a container and are set in the main container file Container.<uid>.xml file.

Table 4-30: MPF parameters

Parameter	Default	Description
omniq.index.mpf.docsPerFile	20	The number of documents stored in each MPF.
omniq.index.mpf.filesPerFolder	250	The number of MPFs stored in each folder.
omniq.index.mpf.foldersPerFolder	50	The number of MPF folders stored per folder.
omniq.index.mpf.maxParagraphGroups	5	The maximum number of paragraph groups to allocate per document.
omniq.index.mpf.maxTotalGroupEntries	50	The maximum number of paragraphs from any one document that can be in a paragraph group.
omniq.index.mpf.bufferSoftLimit	8192	The ideal number of bytes an uncompressed paragraph group can consume before it is closed, compressed, and written to disk. This limit is usually slightly exceeded by design.

CHAPTER 5

Configuring Web Administration

This chapter describes the key configuration parameters for the Hyena servlet container, which is provided as a component of Sybase Search Web Administration. The Hyena servlet container is a standalone lightweight HTTP server for use only with Sybase Search. You can use the Hyena servlet container, or you can integrate Sybase Search with any J2EE application server, such as Apache Tomcat.

Changing the Hyena configuration

The initial Hyena configuration occurs while installing the Web Administration component of Sybase Search. (See "Installing Sybase Search" on page 7.) You can change the configuration of the Hyena servlet container by editing the Hyena configuration file. The Hyena configuration file is *server.xml*, and it is located in *<installLocation>\Hyena\config*. Use a text editor to edit the file.

Table 5-1 shows the attributes for the HTTP server tag.

Table 5-1: HTTP Server tag

Attribute	Default value	Description
port	None	The TCP/IP port on which Hyena listens for connections.
host	*localhost*	The name or IP address of the host on which the Hyena servlet container resides.
stdOutput	false	All standard output (for example, printed to java.lang.System.out and java.lang.System err) is always redirected to the Hyena log file. When set to true, the output is sent to the original standard output (usually the console) as well.

Table 5-2 shows the attributes for the Request-Handler tag.

Table 5-2: Request-Handler tag

Attribute	Default value	Description
minThreads	10	The minimum number of server threads that Hyena uses to serve connections.
maxThreads	75	The maximum number of server threads that Hyena uses to serve connections.
maxIdleTime	10000	The number of milliseconds an idle server thread is kept alive before being destroyed. This parameter applies only when the current number of server threads exceeds the minimum.
debug	false	If set to true, request handling debug information is written to standard output for every HTTP connection received.

Table 5-3 shows the attributes for the Request-Parser tag.

Table 5-3: Request-Parser tag

Attribute	Default value	Description
maxHeaderLength	None	The maximum number of characters accepted in any one HTTP request header (including GET parameters). Requests using headers longer than this are denied. Requests that send large parameter values should use the POST method.
maxNumberOfHeaders	None	The maximum number of request headers accepted as part of any single request. Requests formed using more headers than this are denied.

Table 5-4 shows the attributes for the Request-Keep-Alive tag.

Table 5-4: Request-Keep-Alive tag

Attribute	Default value	Description
enabled	false	When set to true, HTTP keep-alive is used with all HTTP clients that support it.
maxRequests	None	The maximum number of requests that are served by any one connection.
timeout	None	The number of milliseconds the server waits for further requests on an open connection before breaking it.

Table 5-5 shows the attributes for the Remote-Admin tag.

Table 5-5: Remote-admin tag

Attribute	Default value	Description
enabled	false	When set to true, authorized stop and start commands sent via HTTP are accepted.

Attribute	Default value	Description
authCode	None	The authorization code required by the remote administration listener.

Table 5-6 shows the attributes for the Logging tag.

Table 5-6: Logging tag

Attribute	Default value	Description
enabled	false	If set to true, HTTP requests are logged.
directory	None	Designates the directory in which log files are written.
prefix	None	The standard prefix for all log file names (appears before the date).
suffix	None	The standard suffix to use for all log file names (appears after the date).
timestamp	false	If set to true, time of the HTTP request is logged.

Table 5-7 shows the attributes for the container tag.

Table 5-7: Container tag

Attribute	Default value	Description
debug	false	When set to true, servlet/JSP debug information is written to standard output for each request received.

Table 5-8 shows the attributes for the JSP-handler tag.

Table 5-8: JSP-handler tag

Attribute	Default value	Description
vigilance	None	When set to true, servlet/JSP debug information is written to standard output for each request received.

Table 5-9 shows the attributes for the error-template tag.

Table 5-9: Error-template tag

Attribute	Default value	Description
path	<*installLocation*>*config**error_template.htm*	Defines the path to an HTML template with which error messages are formatted to display to clients when an application error is encountered.

Table 5-10 shows the attributes for the context tag.

Table 5-10: Context tag

Attribute	Default value	Description
name	None	All context resource URIs implicitly start with this value; it must begin with a forward slash. For example, the context named */omniq* might have its home page at */omniq/index.html*.
path	None	

The MIME-mapping tag

The MIME-mapping tag defines no attributes but has two other tags nested within each opening and closing pair:

- Extension – its node value represents a file extension, for example, “htm” for HTML documents.
- MIME-type – its node value represents a MIME type, for example, “text/html” for HTML documents.

Use MIME configuration when setting the content-type HTTP response header for requested files.

CHAPTER 6

Customizing Sybase Search

This chapter contains information for developing, configuring, and using custom HTTP handlers, filters, parsers, and text splitters.

Developing and configuring HTTP handlers

An HTTP handler is a Java object designed to service HTTP requests, similar to a simplified Java servlet. Sybase Search allows allocation of any number of HTTP handlers to modules at configuration time. This means you can develop and plug in custom HTTP handlers as necessary. Sybase Search ships with five HTTP handlers, four XML handlers, and a generic file serving handler (for the Document Type Definitions).

See “Configuring the container XML file” on page 45.

XML Document Groups HTTP handler

This handler returns a list of document groups in an XML format compliant with its Document Type Definition (DTD), which can be found at *<installLocation>/OmniQ/config/dtd/DocumentGroups.dtd*.

It lists each document group’s ID for use as a search parameter, as well as its name and the names and addresses of each of the document stores’ members for display and integration purposes.

In a default installation of Sybase Search, you can find the XML handler and its DTD handler respectively at:

- *http://<installLocation>:<container-port>/xml/documentgroups*

- *http://<installLocation>:<container-port>/dtd/documentgroups*

XML metadata HTTP handler

This handler returns a list of all indexable metadata Fields in an XML format compliant with its DTD, which can be found at *<installLocation>/OmniQ/config/dtd/Metadata.dtd*.

This handler lists each Metadata Field internal name (for use as a search parameter) as well as its display name and type for display and integration purposes.

In a default installation of Sybase Search, you can find this XML handler and its DTD handler respectively at:

- *http://<installLocation>:<container-port>/xml/metadata*
- *http://<installLocation>:<container-port>/dtd/metadata*

XML query HTTP handler

This handler takes query parameters over HTTP (GET or POST) and returns a result set in XML compliant with the result set DTD, which can be found at *<installLocation>/OmniQ/config/dtd/ResultSet.dtd*.

In a default installation of Sybase Search, you can find this XML handler and its DTD handler respectively, at:

- *http://<installLocation>:<container-port>/xml/query*
- *http://<installLocation>:<container-port>/dtd/resultset*

The XML query HTTP handler parameters are shown in Table 6-1.

Table 6-1: XML query HTTP handler parameters

Normal query parameters	
terms	A natural language query string describing the concepts that all documents should contain.
notTerms	A natural language query string describing the concepts documents should not contain.
termHi	A value used to indicate whether returned terms should be highlighted. The default is bold. The opening and closing tags are: • termHiTagOpen • termHiTagClose
Linking query parameters	
linkingDocAddr	The address of the document to use to create a `find-similar` query.
Linking query with external document parameters	
targetDSMID	The target Document Store Manager ID. The chosen document store is used by Sybase Search to obtain its initial term statistics when calculating the linking query.
linkingDocPath	The full path to the external document (from the Document Store Manager's perspective) used to create a `find-similar` query.
Common parameters	
documentGroupIds	A comma-delimited list of document group IDs. When present, only documents that are members of these groups are returned.
metadata	A metadata search expression, that takes the form: < *name*><*operator*><*value*> where <*name*> is the internal name of the metadata field; <*operator*> is "=," ">=," or "<=" (the latter two are only supported by numeric and date types) and <*values*> is the criteria of the metadata search.
metadataOpWithinFields	Valid values are AND and OR. • When this parameter is AND and two or more values are presented for any one metadata field, all must match for the query to succeed. • When this parameter is OR, only one of any number of values needs to match for the query to succeed.
metadataOpAcrossFields	Valid values are AND and OR. • When this parameter is AND and two or more metadata parameters are present, all must succeed for the query to succeed. • When this parameter is OR, only one of any number of metadata parameters needs to succeed for the query to succeed.
minDocRel	The minimum relevance, expressed as a percentage, a document must achieve to be included within a result set.
numParas	The number of document excerpts to return for each document returned in the page of results.

Normal query parameters	
scoreUnknownTerms	When set to true, terms present in a query yet unknown to Sybase Search (for example, terms not present in any indexed document) are represented in the scoring algorithm. When set to false, unknown terms are ignored
resultsOffset	An integer value that represents the place in the result set the page of results should begin. The first document in the result-set (for example, the top scoring document) is at offset zero (0). If the offset is greater than the number of results found, Sybase Search returns an empty page of results.
resultsLength	An integer value that represents the number of documents to include in the page of results.
maxResultsNeeded	An integer value that represents the maximum number of results required by the caller (the minimum value is implicitly resultsOffset + resultsLength). This yields performance benefits when queries are cached for returning on a page-by-page basis.
categoryIds	A comma-delimited list of category IDs. When present, only documents that are members of the listed categories are returned.

XML document HTTP handler

This handler returns the text from an indexed document in an XML format compliant with its DTD, which can be found at *<installLocation>/OmniQ/config/dtd/Document.dtd*.

In a default installation of Sybase Search, you can find this XML handler and its DTD handler respectively, at:

- *http://<installLocationt>:<container-port>/xml/document*
- *http://<installLocation>:<container-port>/dtd/document*

The handler accepts the following parameters shown in Table 6-2.

Table 6-2: XML document HTTP handler parameters

Parameter	Description
address	The document address of the document to fetch as XML. The document address format is <DSM-ID>-<DS-ID>-<DOC-ID> (Document Store Manager ID, document store ID, and Document ID).
useParagraphs	When this parameter is true, the body text of the document is broken into paragraphs and formatted between extra paragraph XML tags. If set to false, the entire body text is returned in a large, unbroken block.

XML categories HTTP handler

This handler returns a list of all categories in an XML format compliant with its DTD, which can be found at *<installLocation>/OmniQ/config/dtd/Categories.dtd*.

Each category lists the properties ID, document count, display name, and definition. Definition is the query and metadata parameters used to create it.

In a default installation of Sybase Search, you can find this XML handler and its DTD handler respectively, at:

- *http://<installLocation>:<container-port>/xml/categories*
- *http://<installLocation>:<container-port>/dtd/categories*

Developing and configuring customized filters

Sybase Search uses a third-party solution, Stellent, for parsing many document formats. The Stellent document filter is a multi-filter—in other words, the same filter instance handles all supported MIME types. Thus, the Stellent filter is configured to handle the MIME type */*, indicating that it can filter text from documents of any MIME type presented to it.

When Sybase Search obtains a filter for a document, it first identifies its MIME type from the file extension. For example, *C:\document.pdf* has the MIME type "application" and the subtype "pdf" (application/pdf). Sybase Search then requests a filter from the Filter Factory to handle documents with the identified MIME type.

The filter look-up is performed in this order:

1 If a filter is configured to handle a specific MIME type, that filter instance is returned.

2 If a multi-filter (*/*) is configured, that filter instance is returned.

3 No filter is returned, denoting “not indexable.”

You can add additional filters by editing the XML configuration file located in *<installLocation>\OmniQ\config\FilterFactory.default.xml*. See “Configuring modules” on page 50 for information about the *FilterFactory.default.xml* file.

Developing and configuring customized parsers

This section provides information for Java developers about developing, configuring, and using custom parsers.

Parser classes

You can create custom date, int, and float parsers and plug them into the system.

All parsers implement this common base interface:

```
com.isdduk.text.Parser
     getId() : short
     setId(short) : void
     getName() : java.lang.String
     setName(java.lang.String) : void
     init(com.isdduk.util.map.FastMap): void
```

This interface defines methods that facilitate tracking and displaying information about parser instances loaded in Sybase Search, mainly simple GET and SET methods. There is also an initialization method which takes a map of parameters should the parser require any—this method is guaranteed to be called before parsing commences. You can extend the convenience base class com.isdduk.text.AbstractParser.

Irrespective of whether the convenience base class is utilized, each custom parser class must provide a no arguments constructor and implement the appropriate one of these three specialized parser interfaces:

```
com.isdduk.text.DateParser
     parse(java.lang.String, com.isdduk.util.set.LongSet) : boolean
     parse(java.lang.String) : com.isdduk.util.set.LongSet
     format(long) : java.lang.String

com.isdduk.text.IntParser
     parse(java.lang.String, com.isdduk.util.set.IntSet) : boolean
     parse(java.lang.String) : com.isdduk.util.set.IntSet
     format(int) : java.lang.String

com.isdduk.text.FloatParser
     parse(java.lang.String, com.isdduk.util.set.FloatSet) : boolean
     parse(java.lang.String) : com.isdduk.util.set.FloatSet
     format(float) : java.lang.String
```

The parse method, which returns a Boolean result, should contain the parsing logic; the other parse method should simply create a suitable set object and delegate the call, as it is a convenience method for when there is no suitable set object in its scope. The format method should reverse the parse process and return the date, int, or float value as a string (although this is not always possible).

Note The date parser turns date strings into long values—the number of milliseconds that have passed since the 1st of January 1970 in Coordinated Universal Time (UTC).

Adding the new parser

Once you have compiled the new parser class, you must make it available to the system. The easiest way to do this is by adding the class to a Java Archive (JAR) file and placing the JAR file in the *<installLocation>\OmniQ\lib* directories.

Note You must place the JAR file into every container's library folder.

You must add the new parser to the internal set of parsers. Edit the *<installLocation>\OmniQ\config\Parsers.xml* configuration file and add a new Parser tag.

For example, a parser that parses user IDs from strings might have a configuration similar to this:

```
<Parser identifier="intUserId_8" class="com.mycompany.IntParserImpl">
```

```
     <Param name="base" value="16" />
</Parser>
```

You can load multiple instances of the same class. This example assumes the parser class can parse different integer bases (octal, decimal, hexadecimal, and so on), but the configured instance expects the hexadecimal format.

Note Because Sybase Search is a distributed system, it is important the new parsers are configured for the container instance that loads the Text Manager.

Using the new parser for metadata indexing

You must configure each metadata field that requires the new parser. To do this:

1 Using a text editor, open the *Metadata.xml* configuration file located in *<installLocation>\OmniQ\config*.

2 Locate the appropriate fields and change the parser attribute to the value of the new parser identifier. For example:

```
<Field name="userid" displayName="User ID" type="INT"
parser="intUserId_8" indexable="true" />
```

If it is a new metadata field and does not exist in this file, add it.

Note Because Sybase Search is a distributed system, it is important to perform metadata changes for the container instance that loads the Metadata Manager.

Using the new parser for querying

If the metadata field requires metadata query values to be handled by the new parser, it is necessary to override the default behavior by editing the *QueryParsers.xml* file located in *<installLocation>\OmniQ\config*. If you assume the querying format is the same as it is for indexing, then the query parser would have the following entry:

```
<MetadataField name="userid" parser="intUserId_8" />
```

Note Because Sybase Search is a distributed system, it is important that the metadata changes are performed for the container instance that loads the Query Manager.

Developing and configuring customized text splitters

This section provides information for Java developers about developing, configuring, and using custom text splitters.

All document body text and textual metadata (excluding file paths) values are passed through the configured term splitter to be broken into individual terms. Each term that is not preserved (see "Defining the list of preserved terms" on page 62), not a stopword (see "Defining the list of stopwords" on page 62), and is neither too short nor too long, is passed to the configured term stemmer to be reduced to its root form. Both the term splitter and term stemmer can be reimplemented and reconfigured where necessary.

Term splitting turns extracted plain text into words. Term stemming reduces words to their common roots. Term splitting and term stemming are language-specific; therefore, for optimum performance when you know documents and searches are to be performed in a single language, you can customize the term splitter and term stemmer algorithm to make best use of the language.

For example, an English stemming algorithm converts "singing", "sings", and "singer" to the stem "sing"; however, this algorithm is not appropriate for French or Chinese.

The default configuration splitter class com.isdduk.text.BreakIteratorSplitter handles all double-byte characters by using the underlying default Java class java.text.BreakIterator. The Java *BreakIterator* class uses punctuation and word delimiters to split single-byte languages into words. For double-byte languages; however, the Java *BreakIterator* class samples the glyphs in pairs and tries to determine where the end of the words are likely to be.

If you intend to run Sybase Search with documents containing glyph-based languages, it is recommended that you write your own custom term splitter (described in "Configuring the term splitter"). Term splitting algorithms designed for a single language out-perform the Java *BreakIterator*, which is designed to handle multiple languages, particularly glyph-based languages.

Configuring the term splitter

The term splitter interface defines numerous methods, many of which must be the same, regardless of the splitting algorithm employed. To simplify implementing new term splitters, Sybase Search includes an abstract base class that you can extended to inherit much of the required functionality:

```
com.isdduk.text.AbstractTermSplitter
```

The convenience base class does not implement any splitting algorithms. The various split methods defined by the term splitter interface are as follows (see the Javadocs for the full interface method listing):

```
com.isdduk.text.TermSplitter

     split(java.lang.String source) :
com.isdduk.util.set.FastSet<java.lang.String>
     split(java.lang.String source, boolean validate) :
com.isdduk.text.StringList
     splitFrequencies(char[] source,
          com.isdduk.util.map.FastTermMap insertInto) : void
     splitFrequencies(java.lang.CharSequence source,
          com.isdduk.util.map.FastTermMap insertInto) : void
```

Configuring the term stemmer

The term stemmer interface is much simpler than its splitter counterpart. It defines only three methods:

```
com.isdduk.text.TermStemmer
     stem(com.isdduk.text.Term term) : com.isdduk.text.Term
     hasNormalize() : boolean
     normalize(com.isdduk.text.Term term) : com.isdduk.text.Term
```

The stem method takes a term argument and returns a stemmed version of it, which is in many cases the same object, although perhaps with a different length. The normalize method caters for terms that are not sent through the stem method (which should incorporate normalization as part of its routine)—it ensures the term conforms to a single standard of representation (for example, a German stemmer may normalize the sharp S “ß” to its equivalent “ss” or vice versa). Terms may bypass the stem method occasionally, when their lengths exceed the maximum allowed (and are therefore “force stemmed” to fit).

Replacing the system text and term splitters

After you have compiled the new term splitter class, you must make it available to the system. The easiest way to do this is by adding the class to a JAR file and placing the JAR file in the *<installLocation>\OmniQ\lib* directories.

Note You must place the JAR file into every container's library folder because Sybase Search is a distributed system.

The module responsible for loading the term splitter and stemmer is the Text Manager module. Edit the *TextModule.default.xml* configuration file located in *<installLocation>\OmniQ\config* and change the term.splitter.class property and the term.stemmer.class property as necessary.

Note You must perform the metadata changes for the container instance that loads the Text Manager.

APPENDIX A

Silent Install Mode

Using silent mode to install Sybase Search

Silent install mode is used primarily with the Sybase Data Integration Suite. You can also use silent mode to install Sybase Search if you only have command line access to a server. Silent install mode uses options files to automatically pass parameter settings to the installer. The silent installation process requires no interactive steps.

For silent install mode, Sybase Search provides sample options files named after each installation type. The options files contain configuration parameters that you can set when using silent install mode to install Sybase Search.

Table A-1 on page 92 lists the options files and the configuration parameters relevant to each installation type.

Table A-1: Options files and configuration parameters for silent install

Options file	Parameters
TypicalInstallOptions.txt	installLocation
	IS_SELECTED_INSTALLATION_TYPE
	HUB_CONTAINER_PORT
	CONTAINER_RMI_PORT
	HYENA_PORT
CustomInstallOptions.txt	installLocation
	IS_SELECTED_INSTALLATION_TYPE
	VAR_CHECKBOX_HUB_CONTAINER
	VAR_CHECKBOX_SATELLITE_CONTAINER
	VAR_CHECKBOX_WEB_ADMIN
	CONTAINER_RMI_PORT
	HUB_CONTAINER_IP_ADDRESS
	HUB_CONTAINER_PORT
	CONTAINER_ID
	CONTAINER_PORT
	HYENA_PORT

If you are installing on Windows, refer to the *silent_install.bat* file for examples of the various installation options. If you are installing on Solaris, refer to silent_install.sh for examples of various installation options.

❖ **To install Sybase Search using silent install mode**

1 Verify that you have write permission to the directories on each server where you are installing Sybase Search features.

2 To run the silent installation program, use the platform-specific Sybase Search executable.

- If you are installing on Windows 2000, 2003, or XP, run:

  OmniQ_3.1_win32.exe -is:javaconsole -silent -options *<user-specified options file>*

- If you are installing on Solaris, run:

  OmniQ_3.1_sunsparc64.bin -is:javaconsole -silent -options *<user-specified options file>*

See “Configuration options file for typical install” and “Configuration options file for custom install” to review example configuration options files.

Configuration options file for typical install

The following example shows a sample configuration options file for a typical installation type.

```
################################################
#       Typical Installation
#       (Single-Server Installation)
#
#       Parameters:
#           installLocation
#           HUB_CONTAINER_PORT
#           CONTAINER_RMI_PORT
#           HYENA_PORT
################################################

# The Installation Directory
-P installLocation="C:\Program Files\Sybase\Search3.1"

# The installation type - must be "TypicalInstallation"
# for this options file
-V IS_SELECTED_INSTALLATION_TYPE="TypicalInstallation"

# The port number on which the single container
# will run
-V HUB_CONTAINER_PORT="8001"

# The port number on which the Hub Container RMI service
# will run
-V CONTAINER_RMI_PORT="7000"

# The port number on which the Hyena Web Server
# will run
-V HYENA_PORT="8081"
```

Configuration options file for custom install

The following example shows a sample configuration options files for a custom installation type. Depending on which features you choose to install, you might use all or a combination of these parameters.

```
################################################

#Custom Installation
#         Depending on the choices, will install one
#         or more of:
```

```
#               Hub Container
#               Satellite Container
#               Web Administration Server
#
#          Parameters:
#               installLocation
#               IS_SELECTED_INSTALLATION_TYPE
##################################################
# The Installation Directory
-P installLocation="C:\Program Files\Sybase\Search3.1"

# The installation type - must be CustomInstallation for
# this options file
-V IS_SELECTED_INSTALLATION_TYPE="CustomInstallation"

# The features to be installed. If any are set to "true",
# the parameters below will be read and used,
# otherwise they will be ignored
-V VAR_CHECKBOX_HUB_CONTAINER="false"
-V VAR_CHECKBOX_SATELLITE_CONTAINER="false"
-V VAR_CHECKBOX_WEB_ADMIN="false"

##################################################
#Common Installation Parameters
#
#  Some parameters are shared by each of the custom
#  installation choices.
#  For example, if a user were to install both a Hub
#  and Satellite Containers
#  the Container RMI Port would need to be set to the
#  same value.
#  Likewise, both the Satellite Container and Web
#  Administration Server
#  need to know the Hub Container's server IP address
#
#      Parameters:
#         CONTAINER_RMI_PORT
#         HUB_CONTAINER_IP_ADDRESS
##################################################

# The port number on which the RMI service will run
-V CONTAINER_RMI_PORT="7000"

# The IP address of the Hub Container
-V HUB_CONTAINER_IP_ADDRESS="127.0.0.1"
#
```

```
#
##################################################

# Hub Container Installation
#
#       Parameters:
#          HUB_CONTAINER_PORT
#          CONTAINER_RMI_PORT (defined above)
##########################################################

# The port number on which the hub container will run
-V HUB_CONTAINER_PORT="8001"

#####################################################
#Satellite Container Installation
#
#       Parameters:
#          CONTAINER_ID
#          CONTAINER_PORT
#          CONTAINER_RMI_PORT(defined above)
#          HUB_CONTAINER_IP_ADDRESS (defined above)
#####################################################

# The unique Container ID
-V CONTAINER_ID="2"

# The port number on which the satellite container will
# run
-V CONTAINER_PORT="8002"

################################################
#Web Administration Server Installation
#
#       Parameters:
#          HYENA_PORT
#          HUB_CONTAINER_IP_ADDRESS (defined above)
#          CONTAINER_RMI_PORT (defined above)
################################################

# The port number on which the Hyena will run
-V HYENA_PORT="8081"
```

APPENDIX B

Generated Files

Each module contains its own directory where it stores files. These can be serialized Java object files or proprietary data structures. The format of each directory is the short name of the module followed by its unique module ID.

Module files

Table B-1: Module generated file locations

Module	File location
Document Group Manager	*<installLocation>\OmniQ\data\DGM-<uid>* where *<uid>* is its unique module ID.
Document ID generator	*<installLocation>\OmniQ\data\DocIdGenerator-<uid>* where *<uid>* is its unique module ID.
Filter factory	*<installLocation>\OmniQ\data\FilterFactory-<uid>*
Metadata Manager	*<installLocation>\OmniQ\data\MetadataModule-<uid>* where *<uid>* is its unique module ID.
Metadata Manager Delegate	*<installLocation>\OmniQ\data\MetadataModuleDelegate-<uid>* where *<uid>* is its unique ID.
Query Manager	*<installLocation>\OmniQ\data\QueryModule-<uid>* where *<uid>* is its unique ID.
Repository Module	*<installLocation>\OmniQ\data\RepositoryModule-<uid>* where *<uid>* is its unique ID.
Term Lexicon Manager	*<installLocation>\OmniQ\data\TermLexiconModule-<uid>* where *<uid>* is its unique ID.
Term Lexicon Manager Delegate	*<installLocation>\OmniQ\data\TermLexiconModuleDelegate-<uid>* where *<uid>* is its unique ID.
Text Manager	*<installLocation>\OmniQ\data\TextModule-<uid>* where *<uid>* is its unique ID.

Module	File location
Unique ID generator	*<installLocation>\OmniQ\data\UID-<uid>* where *<uid>* is its unique ID.
Document Store Manager	*<installLocation>\OmniQ\data\DSM-<uid>* where *<uid>* is its unique ID.
Document Stores	*<installLocation>\OmniQ\data\DSM-<uid1>\DS-<uid2>* where *<uid1>* is the unique ID of the Document Store Manager, and *<uid2>* is the ID of the Document Store.

Index

A

C

D

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

X