SYBASE®

Administration and Users Guide

Sybase Search

4.0

DOCUMENT ID: DC35131-01-0400-01

LAST REVISED: May 2008

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

Contents

About This Book

Audience

This guide is for Sybase® Search 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 these chapters:

- Chapter 1, "Retrieving Information Intelligently," describes the features and architecture of Sybase Search.
- Chapter 2, "Administering Sybase Search," includes information on setting up Sybase Search, accessing administration pages, and performing various administration tasks.
- Chapter 3, "Configuring Sybase Search," describes the configuration parameters for containers, modules, and the hub. It includes tips for using the configuration files and changing parameters
- Chapter 4, "Configuring Web Administration," describes the configuration parameters for the Web application provided with Sybase Search.
- Chapter 5, "Customizing Sybase Search," provides information about developing custom filters, parsers, and text tokenizers.
- Chapter 6, "Using Sybase Search," provides information about using Sybase Search to search across documents.
- Appendix A, "Generated Files," gives the names and locations of generated module files.
- Appendix B, "Sybase Search Content Adapter," provides information about installing, configuring, and uninstalling Sybase Search Content Adapter.

Related documents

The Sybase Search documentation set consists of:

- *Sybase Search Release Bulletin for Microsoft Windows, Linux, and UNIX* – contains last-minute information that was too late to be included in the books.
- *Sybase Search New Features* – describes the new features in Sybase Search 4.0.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

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.

Key	Definition
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
sybase\bin	A backward slash ("\") indicates cross-platform directory information. A forward slash ("/") applies to information specific only to UNIX. Directory names appearing in text display in lowercase unless the system is case sensitive.

Accessibility features

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Sybase Search documentation has been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

Note You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool.

For information about how Sybase supports accessibility, see Sybase Accessibility at http://www.sybase.com/accessibility. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

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 network drives, databases, intranets, and the Internet. Sybase Search provides intelligent information retrieval, document index management, and document categorization.

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. Finding and accessing this information often proves difficult when staff or storage rules and structures change.

Sybase Search technology

Sybase Search extracts and processes text content from file systems, databases, intranets, and the Internet. The ability to automatically process unstructured 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 it 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.

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

Sybase Search lets you 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.

Sybase Search features

Sybase Search:

- Supports different formats of data, including most types of document, presentation, spreadsheet, and Web content formats
- Automatically captures and aggregates all unstructured data
- Eliminates preprocessing or manual tagging of files, greatly improving the accuracy and efficiency of document retrieval
- Extracts paragraphs from matching documents
- Finds similar documents by automatically providing a set of relevant content that is conceptually related to each document
- Allows indexing millions of documents using a fully distributed architecture
- Queries and processes using a natural language
- Eliminates any language dependency
- Easily integrates with other applications using Sybase Search well-defined Java APIs, XML APIs, and Web service
- Presents search results with alternative matching, relevant documents that might not have been suggested from the original query
- Supports multidocument files (like XML, ZIP); allows separate documents in a single file to be indexed separately
- Slices large documents into manageable sections that improves search results

Understanding the architecture of Sybase Search

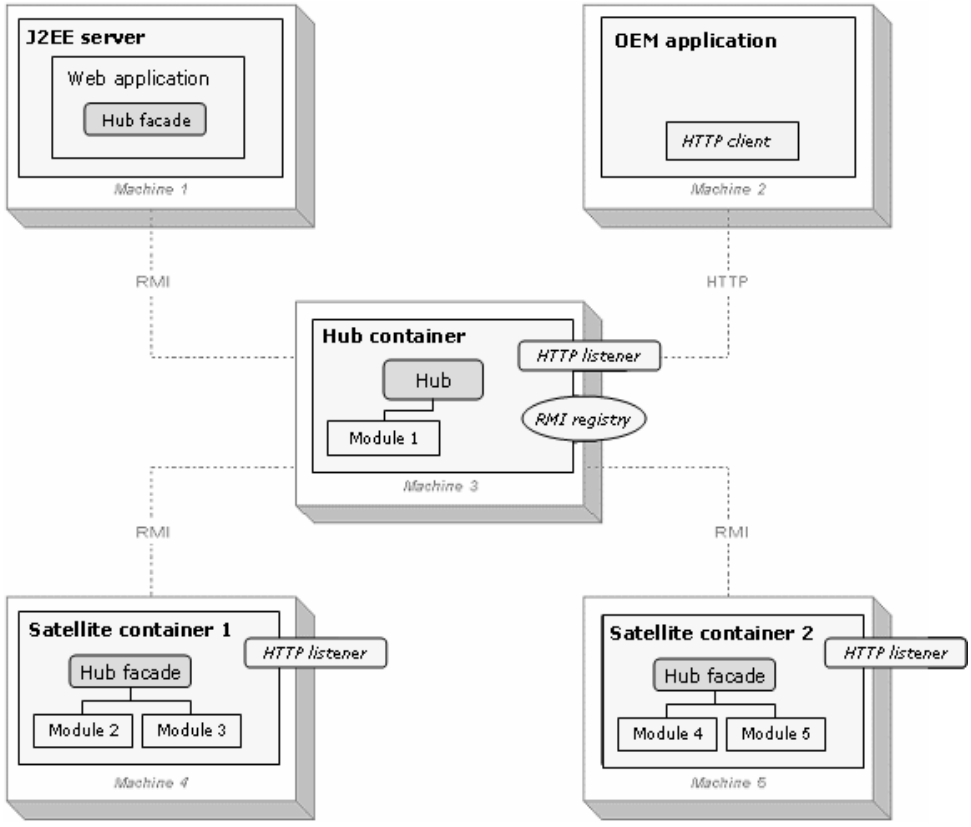
Sybase Search is a fully distributed system, with a central hub server and one or more satellite servers. Each server can contain one or many containers with one or more modules deployed in each container. The exact number of servers, containers, and modules depends on the needs of the installation.

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

For information about the various modules that make up Sybase Search, see Chapter 3, “Configuring Sybase Search.”

Figure 1-1: Sybase Search 4.0 architecture



Optimizing search strategies

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

Optimizing the search engine

A concept-based search engine 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 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 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, it is more common that users are unsure of how to precisely formulate their query, which introduces ambiguity, and less relevant search results. Different vocabulary used to describe similar concepts can also result in important documents being missed, 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.

Querying a number of concepts

Sybase Search is better suited to a query that contains a number of concepts and uses unambiguous 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. An example of a query that is more effective than the AND query, above, 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, commas are used here only for clarity.

This query 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.

Adding variations

It is possible that some relevant documents will still be missed, due to differing vocabulary. Therefore, if you expand the original concepts to include variations that you assume may 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 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 appear at the top of the results 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, try expanding the original query by appending words or phrases from the document search results. The search becomes more accurate as you provide additional information.

Improving relevance

Sybase Search automatically determines the documents that are more relevant than others. This decision is based on the information extracted from all the documents that are indexed by Sybase Search. Part of the relevance calculation assigns an internal weighting for each term in the search query. Depending on the search results, you may want to manually adjust the query term weighting to bias the search results in favor of a particular query term.

For example, Sybase Search has indexed many documents about trains and railway accidents, and incidents. A typical query to find documents about tree branches causing damage to either trains or track:

```
damage being done by tree branches
```

Sybase Search can return relevant documents about damage to branches in the rail tracks that were not caused by trees. This can occur if Sybase Search has indexed documents that are exclusively about "damage to branches in railway tracks," while documents about "tree branches causing damage" include sections about other topics. The second set of documents include relevant matching sections; however, overall, these documents are not as relevant, and are therefore assigned lower relevance scores.

Based on the search results, you can decide to place more emphasis on "tree" damage as opposed to other types of damage. You can use custom term weighting to make your search results more relevant to documents that have references to trees:

```
damage being done by ctw{tree,5} branches
```

Depending on the results from term-weighted search, you can further adjust the custom term weighting to get appropriate emphasis on the term "tree", for example, or any other term to which you want to assign more importance, or a greater weight. See "Setting Text Manager parameters" on page 68 to know about how to use custom term weighting in your queries.

Alternative results for a query

Sybase Search includes query expansion functionality that allows alternative matching for your search results; that is, relevant documents that might not have been suggested from your original query. You can adjust the strength of the query expansion to ensure the search results are either close to the original results or more different.

CHAPTER 2

Administering Sybase Search

This chapter describes how to administer Sybase Search. It includes information on setting up Sybase Search, starting and stopping Sybase Search, accessing administration pages, tracking system details, scheduling tasks, managing documents, categorizing documents, and metadata and language configurations.

Setting up Sybase Search

Before you start Sybase Search for the first time and before any indexing is performed, you must perform:

- Language configuration – decide the languages you want Sybase Search to support.

If Sybase Search is required to work in a language other than English or across multiple languages, you must give consideration to stopwords, text tokenizers, and word stemming configuration. Depending on the languages involved, Sybase Search can provide better results if the stopwords are revised, and the text tokenizer and stemmer are disabled or replaced with custom implementations.

For more information about language-specific configuration of various modules, see "Setting Text Manager parameters" on page 68

For more details on how to write and plug in a new language stemmer into Sybase Search, contact Sybase Technical Support.

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

  For example, if the Query module is located on the hub, review its settings to verify it can handle the required number of concurrent queries.

  For details on configuring various hub-specific modules, see "Modifying module settings" on page 67.

- Satellite container configuration – base this on the level of stress and load you expect of the system. This includes ensuring that the various module caches are set to a high enough level, where memory is available.

  For more details on configuring various satellite container modules, see "Modifying module settings" on page 67 and "Configuring modules using system parameters" on page 83.

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, navigate to:

```
install_location\OmniQ\bin
```

2 Enter:

```
sysearch32.bat -install ID
```

where *ID* is the container ID of the container you want to install as a Windows service. The container ID of the hub container is always 1.

This installs the "Sybase Search – container ID." Use the Microsoft Management Console to run the Sybase Search container as a Windows service.

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

1 From a Windows command prompt, navigate to:

```
install_location\Hyena\bin
```

2 Enter:

```
Hyena32.bat -install
```

This installs the service named "Sybase Search – Web Admin Server." Use the Microsoft Management Console to run Web administration server as a Windows service.

Uninstalling the Windows service

To remove a Sybase Search container as a Windows service:

1 From a Windows command prompt, navigate to:

```
install_location\OmniQ\bin
```

2 Enter:

```
sysearch32.bat -uninstall ID
```

where *ID* is the container ID of the container you want to uninstall as a Windows service. The container ID of the hub container is always 1.

To remove the Sybase Search Web administration server as a Windows service:

1 From a Windows command prompt, navigate to:

```
install_location\Hyena\bin
```

2 Enter:

```
Hyena32.bat -uninstall
```

Starting and stopping Sybase Search

Windows

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

❖ **Starting Sybase Search from the Windows Start menu**

1 Select Programs | Sybase.

2 Select Sybase Search 4.0.

- To start a single server installation, select Start Single Server. The single container and Web administration server starts and runs in a Windows console.
- To start the 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.

❖ **Stopping Sybase Search from the Windows Start menu**

1 Select Programs | Sybase.

2 Select Sybase Search 4.0.

- To stop a single server installation, select Stop Single Server. The single container and Web administration server stops and the Windows console closes.
- To stop the 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.

❖ **Starting Sybase Search from a Windows command prompt**

1 Open a Windows command prompt.

2 To start a container, navigate to *install_location\OmniQ\bin* and enter:

```
sysearch32.bat -start ID
```

where *ID* is the container ID of the container you want to start. The container ID of the hub container is always 1.

Note By default, log files are not created for containers when you use the command line start scripts. To create a log file, use the tee option with the script to start a container, such as:

```
sysearch32.bat -start ID tee
```

3 To start the Web administration server, navigate to *install_location\Hyena\bin* and enter:

```
Hyena32.bat -start
```

❖ **Stopping Sybase Search from a Windows command prompt**

1 Open a Windows command prompt.

2 To stop a container, navigate to *install_location\OmniQ\bin* and enter:

```
sysearch32.bat -stop ID
```

where *ID* is the container ID of the container you want to stop.

3 To stop the Web administration server, navigate to *install_location\Hyena\bin* and enter:

```
Hyena32.bat -stop
```

Solaris and HP

In Solaris and HP, start and stop Sybase Search containers and the Web administration server from a command line.

Note Before starting any containers, set the library path environment variable LD_LIBRARY_PATH. Go to *install_location\OmniQ\bin* and enter:

```
. ./env.sh
```

- To start a Sybase Search container, go to *install_location\OmniQ\bin* and enter:

```
./sysearch64.sh start ID
```

where *ID* is the container ID of the container you want to start.

- To start the Web administration server, go to *install_location\Hyena\bin* and enter:

  ```
  ./Hyena64.sh start
  ```

- To stop a Sybase Search container, go to *install_location\OmniQ\bin* and enter:

  ```
  ./sysearch64.sh stop ID
  ```

  where *ID* is the container ID of the container you want to stop.

- To stop the Web administration server, go to *install_location\Hyena\bin* and enter:

  ```
  ./Hyena64.sh stop
  ```

AIX and Linux

In AIX and Linux, start and stop Sybase Search containers and the Web administration server from a command line.

Note Before starting any containers, set the library path environment variable for the current profile. The library path environment variable is LIB_PATH and LD_LIBRARY_PATH for AIX and Linux, respectively. To set this variable, go to *install_location\OmniQ\bin* and enter:

```
. ./env.sh
```

- To start a Sybase Search container, go to *install_location\OmniQ\bin* and enter:

  ```
  ./sysearch32.sh start ID
  ```

  where *ID* is the container ID of the container you want to start.

- To start the Web administration server, go to *install_location\Hyena\bin* and enter:

  ```
  ./Hyena32.sh start
  ```

- To stop a Sybase Search container, go to *install_location\OmniQ\bin* and enter:

  ```
  ./sysearch32.sh stop ID
  ```

  where *ID* is the container ID of the container you want to stop.

- To stop the Web administration server, go to *install_location\Hyena\bin* and enter:

  ```
  ./Hyena32.sh stop
  ```

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.

Note Before accessing the administration pages, make sure that the Sybase Search has been started and is running properly.

❖ **Accessing the Sybase Search administration pages**

1 Open a Web browser.

2 In the address bar of the Web browser enter:

```
http://hostname:port/omniq
```

where:

- *hostname* is the name or IP address of the machine hosting the Sybase Search Web application.
- *port* is the port number for the J2EE application server hosting the Sybase Search Web application. The default port number is 8111.

3 In the Sybase Search login page, enter the administrator password you provided during the Sybase Search installation and click Login. The Sybase Search Home page appears.

Note If there is no activity in the Sybase Search Web administration page for 30 minutes, the application logs off and you are prompted to log in again to access the Sybase Search Web administration page. To change the login timeout duration, change the value of the session-timeout tag in the *web.xml* file available in the *install_location\webapp\WEB-INF* directory.

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

- Search – lets you search across all 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.
- Configuration – allows you to set up your metadata and language configuration for search optimization. You can add, edit, and remove metadata fields, metadata parsers, synonyms, acronyms, preserved terms, and stopwords.

Tracking system details

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

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

  You can also view the Java system properties of each container's Java Runtime Environment (JRE) and Sybase Search license information.
- Memory Usage – lets you track the memory consumption of the Sybase Search containers, including the Java Virtual Machine (JVM) allocation and consumption. You can also track the resources loaded within the loaded modules, such as data caches.
- Events – 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 a Hub Manager, filtered by severity, date, and sorted in chronological or reverse order.
- Scheduler – 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 in 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" on page 17.

Scheduling tasks

From the System page, you can select Scheduler to configure tasks that run at scheduled time intervals. You can add and edit tasks 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 – deletes old log files and optionally compresses 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 index stripes" on page 38.
- Web robot runner – recrawls the Web sites specified in the Web robot at specified intervals. The Web robot runner does not use the Web robot Force Refresh option.

❖ **Scheduling tasks**

1 From the System page, select Scheduler.

2 Select New Scheduled Task.

3 To enter a label for the task you are scheduling, unselect Auto Label and enter a phrase or short description in the Label field.

  Note By default, Auto Label is selected and a label is automatically created depending on the task you choose to schedule.

4 From the Type list, select a task type. Sybase Search displays fields relevant to the selected task type.

Table 2-1: 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 Nonactive Logs – specify whether you want Sybase Search to compress inactive logs.
Document store runner Index unifier	Document Store – select a document store.
Web robot runner	Web Robot – select a Web robot.

5 In the Every field, enter how often you want the task to run and select one of the options:

- Never
- Minutes
- Hours
- Days
- Weeks
- Months

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

Managing administration roles

Sybase Search allows you to assign a particular role based on the level of access to the search features for different users.

Table 2-2: Preconfigured administrator roles

Role name	Description
SySearch_Admin	Super user – no restrictions
SySearch_ReadOnlyAdmin	Perform queries and all read operations
SySearch_CategoryAdmin	Category and category tree administration
SySearch_FsDocStoreAdmin	File system document store management
SySearch_DbDocStoreAdmin	Database document store management
SySearch_EmDocStoreAdmin	Passive document store management
SySearch_WebRobotAdmin	Web robot management
SySearch_DocGroupAdmin	Document group management

Sybase Search allows you to define new roles or change authorization of existing roles. You can map users to the predefined administrator roles and the associated functionality access in your external system, for example, LDAP.

Managing built-in user accounts

The built-in authentication system in Sybase Search defines two types of user accounts:

- *sysearch_guest* user account – By default, this account allows access only to query functionality. However, you can assign different roles to the guest account. This account does not require a login password.
- *sysearch_admin* user account – allows access to all functionality.

See “Configuring authentication systems” on page 99.

From the Security page you can:

- Modify password for *sysearch_admin* user account
- Enable or disable *sysearch_guest* user account

Note If the Built-in Login module is not installed, Built-in Account page does not display the user accounts.

❖ **Modifying password for administrator account**

1 Click System.

2 Click Security.

3 Select Modify Password for sysearch_admin.

4 Complete these fields:

Field	Description
Old Password	Enter your old account password.
New Password	Enter your new password.
Confirmed Password	Enter your new password again.

5 Click Modify.

❖ **Enabling or disabling the guest account**

1 Click System.

2 Click Security.

3 Select Disable (or Enable) for sysearch_guest.

Managing documents

You can create document stores and organize document stores into groups. You can also create document categories.

Document stores

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

- File system
- Database
- Passive (Web)

The file system and database document stores acquire and maintain documents through internal processes. The Web document stores are passive, and are managed by a Web robot.

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 or mounted remote file systems. 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 (for example, *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 directories and subdirectories. Files with valid MIME types are then indexed. You can customize the list of valid MIME types.

❖ **Creating file system document stores**

1 Click Document Management.

2 Click File System.

3 Click Import from file system.

4 Complete these fields:

Field	Description
Name	Name of the document store.
Manager	Document store manager for which the document store should exist. 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	Document groups in which the document store is a member. See “Grouping document stores” on page 40.
Not Member of	Document groups of which the document store is not a member.

Field	Description
Store Indexed Text	Indicates the raw text from each document is stored within the document store. By default, the option is selected. If you unselect the option, the search results page does not include the View Text link option for each results, as there is no cached text to display. **Note** If you do not want to view the raw text from your documents and have disk space constraints, you can unselect this option.
Index now	Proceeds with indexing immediately or save the configuration without indexing. See "Indexing document stores" on page 35.
Directories	One or more root directories whose contents are indexed and available for searching.
Include subdirectories	Index all subdirectories under the root directory.
File Type Filter	Include or exclude documents by file extension or MIME type, for example: • Include – indexes documents of the specified file type. • Exclude – indexes all documents except those of the specified file type.

5 Click Create.

The Document Stores Summary page shows the details of the document store. An indexing summary is also listed, and, if the store is being indexed, the current indexing session information appears. See "Indexing document stores" on page 35.

Database document stores

A database document store represents a collection of documents imported into Sybase Search from one or more database tables. Use a SQL query to import documents from database tables into Sybase Search. See "Constructing an import SQL statement" on page 26.

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 you can obtain JDBC drivers.

Note The database document stores use Java Database Connectivity (JDBC) drivers to import data. Before creating a database document store, make sure that the appropriate JDBC driver is available in *install_location/OmniQ/lib*. If it is not available, copy an appropriate JDBC driver to *install_location/OmniQ/lib* and restart the container that manages the database import function. For more information about the JDBC driver's location, see your database vendor's documentation.

❖ **Creating database document stores**

1 Click Document Management.

2 Click Database.

3 Click Import from database.

4 Complete these fields:

Field	Description
Name	Name of the document store.
Manager	Document store manager for which the document store should exist. Typically, there is one document store manager for each server where document indexing occurs. The document store manager for each document store lets you set up document indexing on the different servers in the system. See “Managing document stores” on page 39.
Member of	Document groups in which the document store is a member. See “Grouping document stores” on page 40.
Not Member of	Document groups of which the document store is not a member.
JDBC connection details	
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 who has authenticated access to the database.
Password	Indicates the password used to authenticate access to the database.

Field	Description
Preset	Indicates the type of database and the configuration of the JDBC options. When you select a database from the Preset list, Sybase Search automatically displays the port, driver, and URL with common values for the type of database selected. To use a preset database: 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 a preset. 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. If you do not select a preset database from the list, enter driver and URL appropriate values. **Note** Inclusion of a database driver in the Presets list does not mean the driver is available to the system. Make sure that the correct driver is available to the selected document store manager.
Port	Indicates the database server listener port. If you select a database from the Preset list, this field is populated automatically.
Driver	Indicates the full class name of the JDBC driver. If you select a database from the Preset list, this field is populated automatically.
URL	Indicates the JDBC URL to use to contact the database. If you select a database from the Preset list, this field is populated automatically.
SQL Query	Indicates the SQL statement designed to import documents from a database. See “Constructing an import SQL statement” on page 26.
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.

Field	Description
Length	Identifies the document reference length, which is used only for java.lang.String document reference types (the lengths of other types are implicit). In most cases, the value in this field 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.
Store Indexed Text	Indicates the raw text from each document is stored within the document store. By default, the option is selected. If you unselect the option, the search results page does not include the View Text link option for each results, as there is no cached text to display.
Index now	Indicates whether to proceed with indexing immediately or to save the configuration without indexing. See "Indexing document stores" on page 35.

5 Click Create.

The Document Stores Summary page shows the details of the document store. An indexing summary is also listed, and, if the store is being indexed, the current indexing session information appears. See "Indexing document stores" on page 35.

Passive (Web) document stores

A passive document store represents a collection of documents imported into Sybase Search by an external process such as Web robot. The Web robot manages the download of Web content from the Internet and intranets. The Web content is sent to a passive document store, where it is indexed and made available for searching. See "Web robots" on page 30.

Sybase Search also allows you to create custom processes for externally managed document stores. See "Customizing externally managed document stores" on page 123.

Constructing an import SQL statement

You can 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, each document 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 requires multiple queries. If you must specify more than one user name and password or more than one host, more than one SQL query is required, and you must create 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:

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

  Sybase Search supports these 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.

- 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 is 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. DOC_CONTENT_TYPE is not always required, as Sybase Search can automatically detect most common content types. The parameter is not mandatory, and is not required for all TEXT and VARCHAR fields.
- 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, supply a document TITLE, which is shown on the document search results page.

Example

This SQL query shows how a recruitment agency might import the resumes of current candidates:

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

The primary key column ID is used as a document reference, and the document content (body text) is composed from both VARCHAR text and document bytes in a BLOB column. The MIME type of each document is stored in the database. A title is constructed from the first and last name of the candidate, and the preferred salary is saved as metadata.

Example of SQL query with user-defined content type

The next example shows how the content-type detector allows the database import SQL statement to index all the binary data, and makes all these common document formats accessible for searching. For example, if you have MS-Word documents in your table, use:

```
< …
MY_COL AS DOC_CONTENT,
'application/msword' AS DOC_CONTENT_TYPE
…
>
```

This example demonstrates how the DOC_CONTENT_TYPE column can be used to define explicitly the content type for a DOC_CONTENT column.

Editing document stores

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.

❖ **Editing a file system document store**

1 From the File System Document Stores page, select the file system document store.

2 Click Edit. You can change the information in all the fields except Type, ID, and Manager. See "Creating file system document stores" on page 21.

3 Make the changes and click Save Changes.

❖ **Editing a database document store**

1 From the Database Document Stores page, select the database document store.

2 Click Edit. You can change the information in all the fields except Type, ID, Manager, and Length. See "Creating database document stores" on page 24.

3 Make the changes and click Save Changes.

❖ **Editing a passive document store**

1 From the Passive Document Stores page, select the passive document store.

2 Click Edit. You can change the document groups in which the selected passive document store is a member.

3 Make the required changes and click Save Changes.

Removing document stores

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.

❖ **Removing a file system document store**

1 From the File System Document Stores page, select the file system document store.

2 Click Remove.

3 Click OK to confirm the deletion.

❖ **Removing a database document store**

1 From the Database Document Stores page, select the database document store.

2 Click Remove.

3 Click OK to confirm the deletion.

Web robots

Web robots create and manage their own passive document store. A Web robot crawls the specified Web sites and saves the text content of each page locally. However, it does not save the Web content such as images, JavaScript, and style sheets.

Note Web robots may take a considerable time to crawl the target Web sites. Sybase recommends that you configure only one robot per Web site.

❖ **Creating a Web robot**

1 Click Document Management.

2 Click Web Robots.

3 Click Import from the Web.

4 Complete these fields:

Field	Description
Main page	
Name	Name of the Web robot.
Crawl Now	Indicates whether the Web robot should begin crawling immediately, or wait until it is scheduled or manually started later.
Force Refresh	Indicates whether the Web robot should discard the previously collected URL data. When a Web robot crawls a Web site, it stores some of the HTTP response headers of each page it downloads, such as, the *status code*, *Expires*, *Last-Modified*, and *ETag* headers. This information helps to determine whether the page should be downloaded and crawled again. The Force Refresh is selected when you edit the Web robot.
Web Robot Manager	Indicates the Web robot manager that hosts the Web robot.
Passive Document Store Manager	Indicates the Document store manager to which the Web robot should send its documents for indexing.
Store Indexed Text	Indicates whether the raw text from each document is stored within the document store. By default,the option is selected. If you unselect the option, the search results page does not include the View Text link option for each results, as there is no cached text to display.
URLs page	
Start URLs	Indicates the URLs where the Web robot starts crawling.
Link Extractor Patterns	Indicates that the links of the pages downloaded from URLs that match one of these patterns are extracted and put into the URL (work) queue.

Field	Description
Regular Expressions	Indicates whether the patterns should be treated as Java 1.5 regular expressions. A regular expression pattern follows a set of syntax rules to describe or match a set of strings. Go to the Java API Web site at http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/package-summary.html If this option is not selected, patterns are treated as nonregular expressions. Nonregular expression patterns, which begin with http:// or https:// are considered as "starts with" patterns. All other nonregular expression patterns are considered as "contains string" patterns. For example: • http://www.mysite.net – extracts links from all pages. • http://www.mysite.net/public/ – extracts links only from pages in the /public directory. • /public/ – extracts all links that include "/public/" as part of their URL.
Link Extractor Pattern Exceptions	Indicates the exceptions to the general rules specified in Link Extractor Patterns.
Index patterns	Indicates that the pages downloaded from URLs that match one of these patterns are indexed.
Index Pattern Exceptions	Indicates the exceptions to the general rules specified in Index Patterns.
User Agent page	
User-Agent	Corresponds to the HTTP User-Agent request header. This value is sent with all HTTP requests.
Maximum Pages to Download	Indicates the maximum number of pages the Web robot downloads before it terminates and saves what it has crawled.
Maximum Crawl Duration	Indicates the maximum length of time the Web robot spends downloading it terminates and saves what it has crawled. This amount of time may extend into days therefore, you must specify it as an ISO 8601 duration string.
Maximum Consecutive Failures	Indicates the maximum number of consecutive failures the Web robot is allowed before it terminates and saves what it has crawled
Courtesy Timeout	Indicates the length of time, in seconds, the Web robot waits between successful HTTP requests.

Field	Description
Error Timeout	Indicates the length of time, in seconds, the Web robot waits between unsuccessful HTTP requests. Typically, the error timeout is slightly longer than the courtesy timeout, which allows the network and target Web server time to recover before the next attempt.
Maximum Page Tries	Indicates the maximum number of times the Web robot attempts to download any Web page. Set to a higher value to enable robots to overcome temporary network or Web server failures.
Connect Timeout	Indicates the maximum length of time, in seconds, the Web robot waits to connect to the target Web server.
Read Timeout	Indicates the maximum length of time, in seconds, the Web robot waits on a connection to receive a response.
Ignore Robots.txt	*Robots.txt* file contains instructions to prevent Web robots from crawling and indexing certain files and directories on the site.
Authentication page	
HTTP Authentication	
URL (prefix)	Prefix to the URLs that require authentication, for example, http://example.net/protected/
Realm	Indicates the name of the realm, if applicable. A realm is a *database* of user names and passwords that identify valid users of a Web application.
Username	Indicates the user name required for authentication.
Password	Indicates the password required for authentication.
Confirm Password	Reenter the password for confirmation.
Form Authentication	
Action	The URL that performs the authentication. This is the URL to which the HTML form is submitted.
Method	Indicates the request method, either GET or POST.
User name Form Field	
Field Name	Indicates the form input field, which represents the user name, for example, user name, uname, or usr.
Field Value	Indicates the user name value, for example, jsmith.
Password Form Field	
Field Name	Indicates the form input field, which represents the password for example, password, passwd, or pwd.
Field Value	Indicates the password value.

Field	Description
Confirm Password	Reenter the password for confirmation.
Misc. page	
Default Page Names	Indicates the pages names that the Web robot matches with the target Web server's welcome file list, for example, index.html, index.jsp. This enables the Web robot to index only one version: • http://example.net/ • http://example.net/index.html

5 Click Create to create the Web robot.

❖ **Editing a Web robot**

1 Click Document Management.

2 Click Web Robots.

3 Select the Web robot you want to edit.

4 Click Edit. You can change the information in all the fields except Web Robot Manager and Passive Document Store Manager.

5 Click Update when you have finished making changes.

❖ **Removing a Web robot**

1 Click Document Management.

2 Click Web Robots.

3 Select the Web robot you want to remove.

4 Click Remove then confirm that you want to remove the selected Web robot.

5 Click OK. Sybase Search removes the Web robot and the associated passive document store.

Indexing document stores

Indexing involves collecting data about documents owned by a document store and storing their proprietary data structures, generically called indexes. Once the documents in a document store have been indexed, the documents are available for searching.

Data for all documents are 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 indexing sessions can vary dramatically.

When you create a document store, you can select to immediately index. After you have created a document store you can go to the Document Store Information page to perform these type of indexing:

- Incremental Index – re-run the indexing process over the saved document store configuration. All new documents are indexed; all updated documents are reindexed; all deleted documents are removed from the indexes.
- Part Index – define specific documents to add to a document store. There are two types of part indexes:
  - File System Part Index – index only those documents that exist in any of the document store's document roots. If a document is already indexed, Sybase Search checks for changes and reindexes, if necessary. If the document parameter is in the Sybase Search indexes but no longer exists on the file system, it is deleted from the index. The part index process does not check directory trees for new, modified, or deleted documents, which can save a significant amount of time for large document stores.

    Note Documents that are not available in a valid root directory are ignored.

  - Database Part Index – the original SQL statement is re-run to find new, updated, and deleted rows. For large databases, this can be time consuming. However, you can tailor the Database Part Index to fetch only the new and updated rows, or only the document references of the rows that should be removed. It can also accept a delimited list of document references to remove.

  The part index processes are primarily for use within OEM applications.

All data collected during an indexing session is stored in a 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 (specified in the system property omniq.index.buffer.maxMemory) has been exceeded. The buffer shares this memory allocation with the document store's active index stripe. See "Striping index data" on page 37.

Viewing indexed details

To view the index data, click Index Information from the Document Store Information page.

Table 2-3 summarizes the details of indexing activity displayed by a document store. The document store metrics also report the number of document sections added during indexing.

Table 2-3: Indexing activity

Classification	Property	Value
Files	found	The number of files found
	new	The number of new (unindexed) files found
	modified	The number of updated (modified since the previous indexing session) files found
	deleted	The number of files that have been indexed but no longer exist
	attempted	The number of files the indexer attempted to index for which the result is either a success or a failure
	skipped	The number of files purposefully ignored
	unsupported	The number of files that are not supported for indexing
Documents	added	The number of documents added during the current indexing session
	removed	The number of documents removed during the current indexing session
	total	The total number of documents indexed
Sections	added	The number of sections added during the current indexing session
	removed	The number of sections removed during the current indexing session
	total	The total number of sections indexed
	errors	The total number of errors during the current indexing session

Table 2-4 summarizes the data collected during an indexing session. The Document Store Index Information page summarizes the data structures, which is the combination of all completed indexing sessions.

Table 2-4: Index information data

Property	Value
Document sections	The total number of live and deleted document sections in the indexes of all index stripes
Deleted document sections	The total number of deleted document sections in the indexes of all index stripes
Live document sections	The total number of live document sections in the indexes of all document stripes
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 contains exactly zero or one active index stripe. An active index stripe stores all data in memory while keeping a copy on disk for persistence. An active index stripe can always be written to, and, thus may contain data collected over numerous indexing sessions.

When an active index stripe is emptied into a static index stripe, data files from the active index stripe are deleted and the index stripe is discarded. A new active stripe is created the next time an indexing session collects data.

Static index stripes

Each document store's collection of index stripes contains zero or more static index stripes. A static index stripe is a collection of read-only, disk-oriented data structures.

Viewing index stripe information

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

Table 2-5: Index stripe properties

Property	Value
Root directory	The location where the index stripe stores its data
Term Index Segments	The number of segments into which term indexes are divided
Metadata Index Segments	The number of segments into which metadata indexes are divided
Document sections	The total number of live and deleted document sections in the index stripe
Deleted document sections	The number of deleted document sections for which this stripe still holds data (data that is purged on unification)
Live document sections	The number of live document sections for which this stripe holds data
Document lexicon	
Segments	The number of segments into which the document lexicon is divided
Documents (live and deleted)	The number of live and deleted document sections in the lexicon
Document section ID range	The ID range of the document section IDs (first to last)
Last Indexed	The path or reference of the last document indexed and the time it was added **Note** Referred as “path” for file system document stores, a “primary key” for database document stores, and a “UID” for passive document stores.

Unifying index stripes

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

❖ **Unifying an index stripe**

1 From the Document Stores page, select a document store and click Index Information.

2 Click Unify Indexes. This unification process purges data marked for deletion and defragments the indexed data structures.

Dropping indexes

Occasionally, you may need to delete all indexed data for a document store so that the indexes can be rebuilt. For example, if preserved terms or stopwords change, delete and reindex all documents to affect the changes.

❖ **Dropping all index stripes**

1 From the Document Stores page, select a document store and click Index Information.

2 Click Drop Indexes. All indexed data structures for the current document store are removed from the system.

Note If you drop a passive document stores indexes, the external process is not notified. For example, if you drop the indexes of a Web robot's passive document store, you must also edit the Web robot to re-run with force refresh enabled, to ensure all pages are redownloaded and reindexed.

Managing document stores

Depending on the Sybase Search configuration, each container provides a document store manager for each type of document store. For example, a file system import 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.

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

Resizing the query data cache

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

❖ **Resizing the query data cache**

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

2 Click Resize.

3 Enter the value, in megabytes, to increase or decrease the query data cache. The value must be at least 1MB.

4 Click Change.

Grouping document stores

A document group lets you 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 machines; each document store manager has a "resume" document store. You can create a "CV resume" document group to include all three CV resume document stores. You can then use the document group as a search parameter to indicate only the "resume" document group to be searched.

Table 2-6: Document groups properties

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

Document groups

You can create, edit, and remove document groups.

❖ **Creating a document group**

1 Click Document Management.

2 Click Document Groups.

3 Click Create.

4 Enter a name to uniquely identify the document group.

5 Select the document stores that you want to include in the document group from the Document Store Non-Members and click Add.

6 Click Create.

❖ **Editing a document group**

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

2 Make the changes.

3 Click Save Changes.

❖ **Removing a document group**

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

2 Click Remove.

3 Click Yes to confirm the removal.

Categorizing documents

Set up categories to facilitate information retrieval. A category groups documents by content, independent of location or type of document store. 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.

By categorizing documents, you can create groups of documents on behalf of your users. Instead of searching across all documents, you are presented with a predefined set of categories for searching. You can also browse the documents in each category.

Note Large documents are broken into smaller sections in Sybase Search, which are referred to as document sections (or slices) and in such cases the document sections are categorized.

Categories

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

Sybase Search assigns a document to a category if document 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 document relevance helps ensure that the documents in the category are valid matches.

You can also use a category query that consists of only metadata, such as “File Type = HTML”. This creates a category that contains only HTML documents.

You can either search within a category about a certain subject, such as “England World Cup football” or simply use a category to filter search results, such as searching within a category of HTML documents.

Another way to categorize documents is based on the content from one or more training documents. Using train category, Sybase Search extracts the most relevant content from training documents and uses this information as a new internal query to generate matching documents. “Train Category” feature is similar to the “Find Similar” feature where only one document is used as source document except, with category training, relevant content is extracted from more than one document, ensuring that the extracted content is relevant to the training documents. This method has the following benefits:

- Categories are automatically created based on example documents and without a base query.

  For example, a recruitment company wants to create a category based on a sample Java programmer’s resume. Without the ability to create category based on category training, the company would need to produce a category “seed” query, which can vary depending on the individual who was creating the category. With training documents, Sybase Search extracts the most relevant content and creates newly training Java programmer category relevant to the sample resume.

- Retrains a category that was originally created using a seed query. Categories can be trained repeatedly on new training documents to achieve the best results.

For example, creating a category using the seed query "football team" can contain documents on English football or American football, depending on the documents that have been indexed by the system. Retraining this category on a few sample documents about American football ensures that the documents in the category are more relevant.

- Removes the need for manual tagging and continual maintenance of non-trained categories.

❖ **Creating a category using a base query**

1 Click Document Management.

2 Click Categories.

3 Click Create.

4 In the Category Query Terms field, enter a natural-language query. The more information you provide, the more accurate your results are. See "Searching across documents" on page 128.

5 In the Not Terms field, enter terms to indicate concepts dissimilar to those for which you are searching. See "Searching across documents" on page 128.

6 Click the Details tab.

7 Assign a name to distinguish this category from others.

8 Enter text to further describe the category.

9 Click the Document Groups tab.

10 Select one or more document groups to restrict your search.

11 Click the Metadata tab. To include metadata in the category:

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

b 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.

c Enter a value for the metadata parameter. Table 6-1 on page 130 lists the predefined metadata parameters and types.

d If the metadata parameter contains a value that consists of more than one term, select the Within expression operator.

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.

e 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 128.

12 Click the Result Options tab. To set up result options:

a 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 in the category. Documents with scores lower than the percentage that you enter are not included.

b Select the Score Unknown Terms to specify that terms unknown to the system—which, therefore do not exist in any indexed document—be considered by the scoring algorithm.

c Under Training Options, specify the number of results to display per page and the number of paragraphs to display for each document. Select the Term Highlighting to highlight the query terms in the search results.

Note The fields under Training Options assist category training and have no effect on category creation. The values specified in these fields are not saved during category creation.

13 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. The new category and list of relevant documents appears on the View Category page.

❖ **Creating a category using training documents**

1 Perform steps 1 through 12 of "Creating a category using a base query" on page 43.

2 Click Run Category Query. Each search result contains the Add to training documents link.

3 From the search results, determine the training document that matches the information you are searching and click Add to training documents. The name or title of the specified document appears in the Training documents box. You can add up to five training documents.

4 Select Training Documents.

5 Click Train Category. The search result displays documents that fall into the category, sorted by relevance.

6 Click Create.

❖ **Editing a category**

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

2 Click Edit.

3 Make the required changes. See “Creating a category using a base query” on page 43.

4 Click Save.

❖ **Removing a category**

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

2 Click Remove.

3 Click OK to confirm the removal.

Managing the category tree

The category tree allows you to view and organize categories in a tree structure. The categories are listed in the alphabetical order on the category tree. You can rearrange the tree to add a category as a child of another category.

❖ **Organizing categories within the tree**

1 From the Categories page, click Tree.

2 On the category tree, click the category to rearrange.

3 Click Move to.

4 Click the category to which you want to add the selected category as a child.

5 Click Submit.

❖ **Resetting the category tree**

1 From the Category Tree Admin page, click Reset.

2 Click OK to confirm that you want to reset the tree. Sybase Search resets the category tree to the default view, where the categories are listed alphabetically.

Viewing the contents of a document

You can 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 comprises multiple DOC_LINK file system documents, only the first referenced document is returned for viewing.

Documents that have been imported from a Web site contain links to both the original URL and a cached version of the page.

Note The Web robot saves only the text content of each indexed page. Images, JavaScript, and style sheets are not saved. Therefore, a cached Web page may look different from the original Web page.

❖ **Viewing 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 search for documents that contain similar content.

Metadata configuration

You can add new metadata fields and metadata parsers or you can reload metadata fields and metadata parsers from your default XML files. You can also save your custom metadata field and metadata parser configuration as XML files.

Metadata fields

Sybase Search supports these types of metadata fields:

- TEXT – corresponds to any metadata field that contains words or characters. When TEXT metadata values contain numbers or dates, the values are treated as words.

  There are two special, internal text parsers that can be assigned to text metadata fields:

  - TEXT_STANDARD – parses text metadata in the same manner as text content.
  - TEXT_FILENAME – parses document paths.

  TEXT metadata fields do not support range searching

- DATE – 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.

- FLOAT – 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.

- INT – 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 attempts to convert a metadata string value to an integer, while com.isdduk.text.B2KBIntParser parser attempts to convert a metadata value 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.

Different metadata parsers are available for supporting each of these types, which format or modify metadata values in different ways. For example, different DATE parsers parse the date values depending on the format specified.

❖ **Adding new metadata fields**

1 Click Configuration.

2 Click Metadata Fields.

3 Click Add a new metadata field.

4 Complete these fields:

Field	Description
Name	Indicates the name of the metadata field inside the document.
Display Name	Indicates the name to be displayed on the search page.
Type	Indicates whether the metadata type is TEXT, DATE, FLOAT, or INT.
Index Parser	Indicates an existing index parser for the metadata field.
Query Parser	Indicates an existing query parser for the metadata field.
Indexable	Indicates whether the metadata field should be indexed or not.

5 Click Create.

❖ **Editing metadata fields**

1 From the Metadata Configuration Summary page, click Metadata Fields.

2 Click Edit for the field you want to change. You can edit the Display Name, Query Parser, and Indexable field properties.

3 Make the changes and click Save Changes.

❖ **Removing metadata fields**

1 From the Metadata Configuration Summary page, click Metadata Fields.

2 Click Remove for the field you want to delete. You are prompted to confirm whether you want to delete the metadata field.

3 Click OK to confirm the removal.

Metadata parsers

Metadata parsers are used to process metadata values, which are received as strings. Although document body text is processed by the system text tokenizer and stemmer, metadata must often be handled differently, because metadata string values can be numeric and date type.

There are four types of metadata parsers:

- String – supports TEXT type metadata fields
- Numeric decimal – supports FLOAT type metadata fields
- Numeric integer – supports INT type metadata fields
- Date (time) – supports FLOAT type metadata fields

Sybase Search includes these preconfigured metadata parsers – each requires an identifier that consists of two parts, a name and an unique ID.

Table 2-7: Preconfigured parsers

Item	Description
Name	float_1
Class	**com.isdduk.text.SimpleFloatParser** This class parses strings representing decimal numbers into actual decimal numbers. For example, this parser processes the string “3.142” 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, this parser processes both “3” and “3.142” into Java int 3.
Name	dateUK_3
Class	**com.isdduk.text.DateFormatParser**
Name	dateMs1970_4
Class	**com.isdduk.text.Ms1970DateParser**
Parameter	roundTo Value – choose a year, month, day, hour, minute, second, or any other value to indicate that no rounding should take place. This class is a date parser, which 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 (Coordinated Universal Time).
Name	intB2KB_5

Item	Description
Class	**com.isdduk.text.B2KBIntParser** This class parses strings representing byte-size numbers and converts them into kilobyte-sized numbers. For example, the string “2048” (bytes) is parsed as Java int 2 (kilobytes).
Name	datePDF_6
Class	**com.isdduk.text.PDFDateParser**
Parameter	roundTo Value – choose a year, month, day, hour, minute, second, or any other value to denote that 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).
Name	url_7
Class	**com.isdduk.text.URLTermParser** This class splits URL strings into their constituent elements, namely, protocol, host, port, path, extension, and query. Optionally, each element can be indexed separately. The options parameter determines the elements that the parser returns. By default not all elements are not indexed. For example, the values for protocol and port elements, http and 80, respectively, are usually the same for all URLs and hence are not indexed by default.
Parameter	options Value – choose the value that is the sum of the bits that represent the elements the URL parser should return: • PROTOCOL – 1 • HOST – 2 • PORT – 4 • PATH – 8 • EXTENSION – 16 • QUERY – 32 For example, for the parser to return the path and extension URL elements, set the options parameter to 24 (8+16). If you then use this parameter to parse, for example, http://www.mysite.com/about/jobs.html, the parser returns “about”, “jobs”, and “html.”
Name	int2int
Class	**com.isdduk.text.Int2IntParser** This class parses strings representing integer numbers and factors the integer value using operators.

Item	Description
Parameter	• operator Value – can be any these values: • + • - • * • / • factor Value – an integer value that works on the original integer using any of the operators For example, if the operator value is "/" and factor value is "1024" the result is similar in outcome to the B2KBIntParser parser.

❖ **Adding new metadata parsers**

You can create new metadata parsers. The system generates a unique integer ID for each new elements that form part of the parser identifier.

1 Click Configuration.

2 Click Metadata Parsers.

3 Click Add a new metadata parser.

4 Complete these fields:

Name	Description
Parser Name	Name of the parser instance.
Implement Class	Java implementation class.

5 If your metadata parser requires special parameters, click Add, else proceed to step 6. Complete these fields.

Name	Description
Name	The name of the parameter to pass to the parser.
Value	The string value to associate with the parameter name.

6 Click Create.

❖ **Editing metadata parsers**

You can edit metadata parsers only if it is not being used anywhere including in both query parsers and metadata fields that references the metadata parser.

1 From the Metadata Configuration Summary page, click Metadata Parsers.

2 Click Edit for the parser that you want to change.

3 Make the changes and click Save Changes.

❖ **Removing metadata parsers**

You can remove metadata parsers only if it is not being used anywhere including in both query parsers and metadata fields that references the metadata parser.

1 From the Metadata Configuration Summary page, click Metadata Parsers.

2 Click Remove for the parser that you want to delete.

3 Click OK to confirm the removal.

Exporting to and loading from XML

You can save your current metadata fields and metadata parser configurations to an XML file. Similarly, you can load the default configurations and discard your current metadata fields and metadata parser configurations.

❖ **Saving as XML**

1 Click Configuration.

2 Click Save as XML.

3 Click Save.

4 Click OK to confirm that you want to save the configuration information.

Note For data integrity purposes, the original XML file is not overwritten. Sybase Search exports the metadata field and metadata parser configuration to an XML file available at the following location, which gets overwritten every time you save your settings:

install_location\OmniQ\data\config\Metadata.106.xml, and

install_location\OmniQ\data\config\Parsers.106.xml.

❖ **Loading from XML**

1 Click Configuration.

2 Click Load from XML.

3 Click Load.

4 Click OK to confirm that you want to load the default configurations. The metadata fields and metadata parser configurations are loaded from the default XML files available at: *install_location\OmniQ\config\Metadata.xml*, and *install_location\OmniQ\config\Parsers.xml*.

Warning! If the metadata fields and metadata parsers have been changed from the default configuration, and documents have been indexed using the new metadata fields and metadata parsers, reverting to the default configuration might have an adverse effect on the system.

Language configuration

You can configure synonyms, acronyms, preserved terms, and stopwords. The use of synonyms and acronyms collectively, in Sybase Search, is called **query augmentation**.

Synonyms

Synonyms are implemented as list of words that are considered to have the same meaning:

- 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 previous synonym examples, becomes the following query:

```
The medicine made me drowsy, lethargic, listless,
```

```
sleepy.
```

Note Synonyms are not applied to TEXT metadata fields like “title” or “author”.

❖ **Adding a synonym**

1 Click Configuration.

2 Click Synonyms.

3 Click Add Synonyms.

4 Enter two or more words in the Synonyms field.

5 Click Add if you have multiple entries for a synonym, else proceed to step 6.

6 Click Save.

❖ **Editing a synonym**

1 From the Metadata Configuration Summary page, click Synonyms.

2 Click Edit for the synonym you want to change.

3 Make the changes and click Save Changes.

❖ **Removing a synonym**

1 From the Metadata Configuration Summary page, click Synonyms.

2 Click Remove for the synonym you want to delete.

3 Click OK to confirm the deletion.

❖ **Saving as XML**

1 From the Metadata Configuration Summary page, click Synonyms.

2 Click Save as XML.

3 Click Save, then confirm whether you want to save your current synonyms and acronyms.

Note Sybase Search exports acronyms together with synonyms to an XML file. For data integrity purposes, the original XML file is not overwritten. Sybase Search writes the new settings into an XML file available at the following location:

install_location\OmniQ\data\config\locale\QueryAugmentor_en.104.xml

4 Click OK.

❖ **Loading from XML**

1 From the Metadata Configuration Summary page, click Synonyms.

2 Click Load from XML.

3 Click Load, then confirm whether you want to load synonyms and acronyms from your XML file.

4 Click OK.
The synonyms are loaded from the default XML file available at: *install_location\OmniQ\config\locale\QueryAugmentor_en.xml*.

Note After you load synonyms and acronyms from the XML file, re-run Categories by doing a "Refresh All" from the Categories page to take into account the configuration changes to ensure that the category and View Category results are in synchronization.

Acronyms

Acronyms are implemented as a single acronym key with one or list of corresponding phrase values. In the following example the key "USA" is associated to the phrase "United States of America", and the key "DVD" is associated to both the phrases "Digital Versatile Disc" and "Digital Video Disc":

- USA = United States of America
- DVD = Digital Versatile Disc, Digital Video Disc

Acronyms can augment a query in two ways:

- Acronym expansion - when an acronym key is found in a user's search terms, all the corresponding phrases are added to the original query.
- Acronym resolution - when an acronym phrase is found in a user's search terms, the corresponding key is added to the original query.

Note Acronyms are not applied to TEXT metadata fields.

❖ **Adding an acronym**

1 Click Configuration.

2 Click Acronyms.

3 Click Add Acronym.

4 Complete these fields:

Name	Description
Acronym	Indicates the acronym
Phrase	Complete expansion of the acronym

5 Click Add if you have multiple phrases for a single acronym, else proceed to step 6.

Note Your phrase must have two valid words or else Sybase Search does not allow you to add the new acronym.

6 Click Save.

❖ **Editing an acronym**

1 From the Metadata Configuration Summary page, click Acronyms.

2 Click Edit for the acronym you want to change.

3 Make the changes and click Save Changes.

❖ **Removing an acronym**

1 From the Metadata Configuration Summary page, click Acronyms.

2 Click Remove for the acronym you want to delete.

3 Click OK to confirm the deletion.

❖ **Saving as XML**

1 From the Metadata Configuration Summary page, click Acronyms.

2 Click Save as XML. The Save as XML page appears.

3 Click Save, then confirm whether you want to save your current synonyms and acronyms.

Note Sybase Search exports acronyms with synonyms to an XML file. For data integrity purposes, the original XML file is not overwritten. Sybase Search writes the new settings into an XML file available at the following location:

install_location\OmniQ\data\config\locale\QueryAugmentor_en.104.xml

4 Click OK.

❖ **Loading from XML**

1 From the Metadata Configuration Summary page, click Acronyms.

2 Click Load from XML.

3 Click Load, then confirm whether you want to load synonyms and acronyms from your XML file.

4 Click OK.
The acronyms are loaded from the default XML file available at: *install_location\OmniQ\config\locale\QueryAugmentor_en.xml*.

Note After you load synonyms and acronyms from the XML file, re-run Categories by doing a "Refresh All" from the Categories page to take into account the configuration changes to ensure that the category and View Category results are in synchronization. Also, both acronyms and synonyms are stored in the same file, so you are not allowed to load one without loading the other.

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 query performance.

❖ **Adding a stopword**

1 Click Configuration.

2 Click Stopwords.

3 Click Add Stopword.

4 Enter your terms in the Stopwords field.

5 Click Add if you have multiple entries for a stopword, else proceed to step 6.

6 Click Save.

Note The new stopwords affect only new documents. Previously indexed documents are not affected by new stopwords terms.

❖ **Removing a stopword**

1 From the Metadata Configuration Summary page, click Stopwords.

2 Click Remove for the stopword you want to delete.

3 Click OK to confirm the deletion.

❖ **Saving as XML**

1 From the Metadata Configuration Summary page, click Stopwords.

2 Click Save as XML.

3 Click Save, then confirm whether you want to save your current stopwords terms.

Note For data integrity purposes, the original XML file is not overwritten. Sybase Search writes the new settings into an XML file available at the following location:

install_location\OmniQ\data\config\locale\Stopwords_en.104.xml.

4 Click OK.

❖ **Loading from XML**

1 From the Metadata Configuration page, click Stopwords.

2 Click Load from XML.

3 Click Load, then confirm whether you want to load stopword from your XML file.

4 Click OK.
The preserved terms are loaded from the default XML file available at: *install_location\OmniQ\config\locale\Stopwords_en.xml*.

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" is removed from any extracted text if you entered "us" in the list of preserved terms. The case-sensitive list of preserved terms ensures that "us" is removed, but "US" is indexed and made available to the query calculations.

❖ **Adding a preserved term**

1 Click Configuration.

2 Click Preserved Terms.

3 Click Add Preserved Term.

4 Enter your terms in the Preserved Terms field.

5 Click Add if you have multiple entries for a preserved term, else proceed to step 6.

6 Click Save.

Note The new preserved terms affect only index documents. Previously indexed documents are not affected by new preserved terms.

❖ **Removing a preserved term**

1 From the Metadata Configuration Summary page, click Preserved Terms.

2 Click Remove for the preserved term you want to delete.

3 Click OK to confirm the deletion.

❖ **Saving as XML**

1 From the Metadata Configuration Summary page, click Preserved Terms.

2 Click Save as XML.

3 Click Save, then confirm whether you want to save your current preserved terms.

Note For data integrity purposes, the original XML file is not overwritten. Sybase Search writes the new settings into an XML file available at the following location:

install_location\OmniQ\data\config\locale\PreservedTerms_en.104.xml.

4 Click OK.

❖ **Loading from XML**

1 From the Metadata Configuration page, click Preserved Terms.

2 Click Load from XML.

3 Click Load, then confirm whether you want to load preserved terms from your XML file.

4 Click OK.
The preserved terms are loaded from the default XML file available at: *install_location\OmniQ\config\locale\PreservedTerms_en.xml*.

CHAPTER 3

Configuring Sybase Search

This chapter describes the configuration parameters for containers, the hub, and modules. It includes tips on how to change parameters in the configuration files.

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 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:

```
<Container id="1" port="7701"
  <SystemProperty name="exampleName"
  value="exampleName"/>
  <Hub local="true" host="127.0.0.1" port="7700"
  bindName="Hub" logEvents="true"/>
  <Module id="101"
class="com.omniq.xmp.ExampleModule"
  name="Example Module"/>
```

```
  <Data directory="G:\example\data"/>
</Container>
```

Sybase Search containers embed an HTTP server container, which allows you to support a large number of HTTP handlers. 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>
```

See “Developing and configuring HTTP handlers” on page 109.

The hub

The hub is a special module that is the global coordinator of Sybase Search. The container that loads the hub also runs a Java Remote Method Invocation (RMI) registry to listen for remote requests. Satellite containers load a hub facade that handles communication with the real hub. All queries and administration requests are negotiated by the hub.

Configuration and ID conventions

To obtain example configuration files, install the required container.

The *install_location\OmniQ\config\Container.1.xml* file contains single-server configuration.

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

The files for multiple-server configuration are:

- *install_location\OmniQ\config\Container.1.xml*
- *install_location\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 10,000 by default. See "Setting Unique ID (UID) Generator parameters" on page 67.

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, Sybase recommends that you employ a numbering convention. For example, you might use these conventions in a two-server configuration:

- Container ID – a value from 1– 99. If your installation requires more than 99 servers, a different convention is required.
- 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 7701. For example, the port is 7701 for container 1 and 7702 for container 2.
- Hub container – always binds the RMI registry on port 7700.
- Hub facade ID – on satellite containers the hub facade ID is 100 times the container ID. For example, the hub facade ID for container 2 is 200.

  Note The default Web application always allocates its hub facade ID as 999 as it is not required 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.

Table 3-1 shows the attributes for the container tag.

Table 3-1: Container tag attributes

Attribute	Default value	Description
id	None	The unique ID of the container. This value identifies 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 3-2 shows the attributes for the SystemProperty tag. The system properties include JVM settings and global indexing and querying parameters for modules loaded within the container.

Table 3-2: SystemProperty tag attributes

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 3-3 shows the attributes for the hub tag.

Table 3-3: Hub tag attributes

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	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.
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	Indicates whether the event log should be enabled. The location of the hub is irrelevant.
logDirectory	*data.directory\log*	The full path of the directory in which events logs should be written. If logEvents is false, this attribute is not required.

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

Table 3-4: Deployment tag attributes

Attribute	Default value	Description
Deployment passwordHash	None	Indicates a hash of the deployment password chosen during installation. This value must be identical for all containers participating in a Sybase Search deployment, as it is used for intercontainer authentication.

Table 3-5 shows the attributes for the module tag.

Table 3-5: Module tag attributes

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 3-6 shows the attributes for the HttpHandler tag.

Table 3-6: HttpHandler tag attributes

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.

Deployment password

During Sybase Search installation you must mandatorily specify the deployment password for each container. The deployment password is hashed and stored in the container's configuration file. The deployment password is used by the satellite containers to communicate with the hub container, thereby bypassing user security checking, which improves the performance.

Note The deployment password must be the same for every hub and satellite container, or else the satellite container cannot connect to the hub container.

Modifying module settings

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:

- Unique ID (UID) Generator
- Document Group Manager
- Text Manager
- Term Lexicon Manager
- Term Lexicon Manager Delegate
- Metadata Manager
- Metadata Manager Delegate
- Query Manager
- Repository Manager
- Document Filter
- Category Manager
- Database Import Manager
- File System Import Manager
- Passive Import Manager
- Web Robot Manager
- Category Tree Manager

Setting Unique ID (UID) Generator parameters

The Unique ID Generator settings are loaded through the *UIDGeneratorModule.default.xml* configuration file.

Table 3-7: UIDGeneratorModule.default.xml parameters

Parameter	Default	Description
filename	uid.dat	The file that stores the next unique ID.

Parameter	Default	Description
alwaysOpen	false	If set to true, the underlying Java class leaves the file handle open to the file name above.
seed	10,000	The UID generator seed starts from 10,000; because numbers less than 10,000 are reserved by Sybase Search as module IDs.

Setting Document Group Manager parameters

The Document Group Manager 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 settings are loaded through the *TextModule.default.xml* configuration file.

Table 3-8: TextModule.default.xml parameters

Parameter	Default	Description
acronym.suf.filename	*acronym.ser.suf*	Indicates the location where your system stores updates to your acronyms.
custom.term.weight.tag.start	ctw{	Indicates the start of the custom term weighting parameter among search terms. For example, the format for a complete parameter, for the search term "Sybase" with the weighting increased 5 times, is ctw{Sybase,5}.
custom.term.weight.tag.delim	,	The delimiter used to separate search terms from the custom weight.
custom.term.weight.tag.end	}	Indicates the end of a custom term weighting parameter.
custom.synonym.tag.start	syn{	Indicates the start of a query synonym among search terms. For example, the format for a query to find a manager named Joseph or Joe, is `"Manager syn{Joseph,Joe} Williams"`.
custom.synonym.tag. delim	,	The delimiter used to separate search terms.
custom.synonym.tag.end	}	Indicates the end of the query synonym.
min.term.length	2	The minimum term length considered for indexing. The parameter is not used for preserved terms and does not apply to single-digit terms.

Parameter	Default	Description
max.term.length	20	The maximum term length considered valid for indexing. This value must match the Term Lexicon Manager parameter term.length.max.
para.minVTC	50	Indicates the minimum valid term count for breaking the paragraph text.
para.maxVTC	100	Indicates the maximum valid term count for breaking the paragraph text.
para.maxChars	1500	The maximum characters is used to force-break a paragraph before the para.minVTC has been reached. This ensures that bad text data does not result in big paragraphs.
parsers.filename	*Parsers.xml*	The name of the file in the *config* directory that contains the list of text parsers.
preserved.terms.filename	*locale/PreservedTerms_en.xml*	Contains a 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 “Preserved terms” on page 59.
preserved.terms.suf.filename	*preserved.terms.ser.suf*	Indicates the location where your system stores updates to your preserved terms.
query.augmentor.filename	*locale/QueryAugmenter_en.xml*	Contains a list of synonyms and acronyms.
query.augmentor.verboseLoad	false	Set to true to log details of cases where configurations cannot be strictly adhered to. For example, if the synonyms “transport” and “transportation” are provided, the QueryAugmentor creates a log stating that “transportation” collapses to “transport,” so the synonyms are not loaded.
slice.idealVTC	2000	The ideal valid term count for slicing documents.
stopwords.filename	*locale/Stopwords_en.xml*	Contains a list of stopwords to ignore during the indexing and querying processes to improve system performance. See “Stopwords” on page 58.
stopwords.suf.filename	*stopwords.ser.suf*	Indicates the location where your system stores updates to your stopwords.
synonym.suf. filename	*synonym.ser.suf*	Indicates the location where your system stores updates to your synonyms.

You can set the text tokenizer and stemmer classes to language-independent classes or to language-specific classes. Language-specific stemmers improve system performance when Sybase Search indexes documents only in one language.

Setting text tokenizer parameters

The text tokenizer parameters are loaded through the *TextModule.default.xml* configuration file in the hub container. Table 3-9 shows the configurable attributes for the TextProcessor tag in this file.

Table 3-9: Text tokenizer parameters

Parameter	Default	Description
TextTokenizer	com.isdduk.text.parsing.StdTextTokenizer	Defines the class name of the text tokenizer being used in the whole system.
TermStemmer	com.isdduk.text.Porter2Stemmer	Defines the class name of the term stemmer being used in the whole system.
Param	None	Defines the parameters being used by the local text tokenizer.

Satellite containers use the same class defined in the *TextModule.default.xml* configuration file, so you need not define the class name of the text tokenizer.

Setting Term Lexicon Manager parameters

The Term Lexicon Manager settings are loaded through the *TermLexiconModule.default.xml* configuration file.

Table 3-10: Term Lexicon Manager parameters

Parameter	Default	Description
term.length.max	20	The maximum term length considered valid for indexing. This value must match the Text Manager parameter max.term.length.
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. Idle time restarts 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, set this parameter to the same value as the term.length.max.

Parameter	Default	Description
minimization.factor	50	The branching factor of the underlying term lexicon segments. **Warning!** 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

This module is for use on the containers that do not host the Term Lexicon Manager. The Term Lexicon Manager Delegate is a cache of terms and their unique IDs. Each time the Term Lexicon Manager Delegate fetches a value from the Term Lexicon Manager, the value is cached. This reduces the amount of RMI communication between the two hosting containers.

The Term Lexicon Manager Delegate settings are loaded through the *TermLexiconModuleDelegate.default.xml* configuration file.

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

Setting Metadata Manager parameter

The Metadata Manager settings are loaded through the *MetadataModule.default.xml* configuration file.

Table 3-11: MetadataModule.default.xml parameters

Parameter	Default	Description
metadata.suf.filename	*Metadata.ser.gz*	Indicates the location where your system stores the updates to your metadata fields.
metadata.config.filename	*Metadata.xml*	Contains a list of metadata fields for indexing.
metadata.uid.filename	*metadata.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.
parser.suf.filename	*parser.ser.suf*	Indicates the location where your system stores the updates to your parser.
parser.config.filename	*Parsers.xml*	Contains a list of parsers processing metadata values.
parser.uid.filename	*parser.uid.dat*	The name of the file that stores the next unique ID, which is used when creating new parser. Do not change this parameter.

Setting Metadata Manager Delegate parameters

This module is for use on the containers that do not host the Metadata Manager. The Metadata Manager Delegate is a cache of terms and their unique IDs. Each time the Metadata Manager Delegate fetches a value from the Metadata Manager, the value is cached. This reduces the amount of RMI communication between the two hosting containers.

The Metadata Manager Delegate settings are loaded through the *MetadataModuleDelegate.default.xml* configuration file.

Table 3-12: Metadata Manager Delegate parameter

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.

Setting Query Manager parameters

The Query Manager settings are loaded through the *QueryModule.default.xml* configuration file.

Table 3-13: 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.

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 Document Filter parameters

The Document Filter parameters are loaded through the *DocumentFilters.xml* configuration file.

The default filters in the configuration file are:

- HTML filter
- Plain Text filter
- PDF filter
- POI filter
- RFC822 filter
- RTF filter
- XMLInputMatching filter
- ImportXML filter
- Streaming API for XML (StAX) filter
- ZIP filter

Each filter specifies which class is loaded for the filter. In addition, during the installation of Sybase Search Content Adapter, *DocumentFilters.xml* adds these two filters:

- SearchML
- SearchMLExport

See “Setting Document Filter parameters for Content Adapter” on page 145.

Table 3-14: Document Filter parameters

Parameter	Default	Description
DocumentFilter class	None	The Java class that defines the filter.
Timeout millis	45,000	Indicates the time, in milliseconds, the filter waits while filtering a document. If the filter exceeds the given time, the filter aborts.
TempFiles keep	false	If set to true, the filter keeps any temporary files produced during the filtering process.

Parameter	Default	Description
FallbackCharset	None	Indicates the character set decoding scheme to use for decoding the text bytes when the encoding is not supplied and cannot be determined. **Note** Document filters that implement the com.omniq.flt.TextDocumentFilter interface can use this parameter.

DocumentFilters.xml specifies the MIME type association with each filter within the MimeMapping tag.

Table 3-15: MimeMapping parameters

Parameter	Default	Value
MimeType	None	Specifies the MIME type associated with the filter.
DocumentFilterName	None	Specifies the name of the filter that maps to the MIME type.

Setting the HTML filter parameters

The HTML filter parses HTML files.

Table 3-16: HTML filter parameters

Parameter	Value
DocumentFilter class	com.omniq.filter.html.HTMLFilter
FallbackCharset	Default

Table 3-17 shows the MimeMapping parameters associated with the HTML filter.

Table 3-17: MimeMapping parameters for HTML filter

Parameter	Value
MimeType	text/htm
DocumentFilterName	HTML

Setting the PlainText filter parameters

Use the PlainText filter to parse plain text files. Plain text files seldom contain any information about how they are encoded, so the text filter is often forced to use a default decoder when the code is not known.

If you do not define FallbackCharset property, the filter uses the code identified by the standard Java system property file.encoding.

Table 3-18: PlainText filter parameter

Parameter	Value
DocumentFilter class	com.omniq.filter.txt.PlainTextFilter
FallbackCharset	Default

Table 3-19 shows the MimeMapping parameters associated with the PlainText filter.

Table 3-19: MimeMapping parameters for PlainText filter

Parameter	Value
MimeType	text/plain
DocumentFilterName	PlainText

Setting the PDF filter parameters

The PDF filter parses PDF files.

Table 3-20: PDF filter parameter

Parameter	Value
DocumentFilter class	com.omniq.filter.pdfbox.PDFBoxFilter

Table 3-21 shows the MimeMapping parameters associated with the PDF filter.

Table 3-21: MimeMapping parameters for PDF filter

Parameter	Value
MimeType	application/pdf
DocumentFilterName	PDF

Setting the POI filter parameters

The POI filter parses Microsoft Word, Excel, and PowerPoint documents.

Table 3-22: POI filter parameter

Parameter	Value
DocumentFilter class	com.omniq.filter.poi.POIFilter

Table 3-23 shows the MimeMapping parameters associated with the POI filter.

Table 3-23: MimeMapping parameters for POI filter

Parameter	Value
MimeType	application/msword application/vnd.ms-excel application/vnd.ms-powerpoint application/oda
DocumentFilterName	POI

Setting the RFC822 filter parameters

The RFC822 filter parses RFC822 files, a format commonly used by e-mail systems.

Table 3-24: RFC822 filter parameter

Parameter	Value
DocumentFilter class	com.omniq.filter.rfc822.RFC822Filter

Table 3-25 shows the MimeMapping parameters associated with the RFC822 filter.

Table 3-25: MimeMapping parameters for RFC822 filter

Parameter	Value
MimeType	message/rfc822
DocumentFilterName	RFC822

Setting the RTF filter parameters

The RTF filter parses RTF files. Table 3-26 shows the parameter settings used by the RTF filter.

Table 3-26: RTF filter parameter

Parameter	Value
DocumentFilter class	com.omniq.filter.rtf.RTFFilter

Table 3-27 shows the MimeMapping parameters associated with the RTF filter.

Table 3-27: MimeMapping parameters for RTF filter

Parameter	Value
MimeType	application/rtf
DocumentFilterName	RTF

Setting the XMLInputMatching filter parameters

The XMLInputMatching filter parses XML files by either matching source XML format with Sybase Search import XML format, or by transforming the source XML format; using a custom XSL style sheet. XMLInputMatching filter then redirects the unmatched XML to the StAX filter.

Table 3-28: XMLInputMatching filter parameters

Parameter	Value
DocumentFilter class	com.omniq.filter.xml.XMLInputMatchingFilter
InputMatching Parameters for matching ImportXML by namespace	
InputMatcherClass	com.omniq.flt.xml.NamespaceMatcher
Pattern	urn:schemas-sybase-com/sysearch-import
Action type	redirect
Params:	
Name	DocumentFilterName
Value	ImportXML
InputMatching Parameters for matching Import XML by root element	
InputMatcherClass	com.omniq.flt.xml.RootElementMatcher
Pattern	Import
Action type	redirect
Params:	
Name	DocumentFilterName ImportXML
Value	ImportXML
InputMatching Parameters for fallback	
InputMatcherClass	com.omniq.flt.DefaultInputMatcher
Action type	redirect
Params:	
Name	DocumentFilterName
Value	StAX

Table 3-29 shows the MimeMapping parameters associated with the XMLInputMatching filter.

Table 3-29: MimeMapping parameters for XMLInputMatching filter

Parameter	Value
For applications:	
MimeType	application/xml
DocumentFilterName	XMLInputMatcher
For text:	
MimeType	text/xml
DocumentFilter	XMLInputMatcher

Setting the ImportXML filter parameters

The ImportXML filter parses XML files that are formatted as per Sybase Search Import XML schema.

Table 3-30: ImportXML filter parameter

Parameter	Value
DocumentFilter class	com.omniq.flt.xml.ImportXMLFilter

Table 3-31 shows the MimeMapping parameters associated with the ImportXML filter.

Table 3-31: MimeMapping parameters for ImportXML filter

Parameter	Value
MimeType	application/xml
DocumentFilterName	ImportXML

Setting the StAX filter parameters

The StAX filter parses all types of XML files. The filter treats all tag content as body text and ignores the tag attributes. If you need not fine-tune your XML indexing, use the StAX filter.

Table 3-32: StAX filter parameter

Parameter	Value
DocumentFilter class	com.omniq.flt.xml.StAXFilter

Table 3-33 shows the MimeMapping parameters associated with the StAX filter.

Table 3-33: MimeMapping parameters for StAX filter

Parameter	Value
MimeType	application/xml
DocumentFilterName	StAX

Setting the ZIP filter parameters

The ZIP filter parses ZIP files.

Table 3-34: ZIP filter parameters

Parameter	Value
DocumentFilter class	com.omniq.flt.zip.ZipFilter
tempBufferSize	64kb
detectorClassName	com.isdduk.io.Magic
detectorReadLimit	512
TempFiles keep	false

Table 3-35 shows the MimeMapping parameters associated with the ZIP filter.

Table 3-35: MimeMapping parameters for ZIP filter

Parameter	Value
MimeType	application/zip
DocumentFilterName	ZIP

Setting Category Manager parameters

The Category Manager settings are loaded through the *CategoryModule.default.xml* configuration file.

Table 3-36: Category Manager parameters

Parameter	Default	Description
categoryRunnerPool.size	20	The number of concurrent threads used to run category queries.
reservedNames	Category or Categories	Reserved names that cannot be used as category names when you create categories.
reservedNames.isCaseSensitive	false	Indicates whether reserved names are case-sensitive or not.

Setting Database Import Manager parameters

The Database Import Manager settings are loaded through the *DBDocumentStoreModule.default.xml* configuration file.

Table 3-37: Database Import Manager parameters

Parameter	Default	Description
cache.queryData.capacityInBytes	52428800	The maximum amount of memory allowed for document store manager query data cache.
database.config.filename	DBConfig.xml	The name of the database presets XML configuration file.

Setting File System Import Manager parameters

The File System Import Manager settings are loaded through the *FSDocumentStoreModule.default.xml* configuration file.

Table 3-38: 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.

Setting Passive Import Manager parameters

The Passive Import Manager settings are loaded through the *EMDocumentStoreModule.default.xml* configuration file.

Table 3-39: Passive 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.

Setting Web Robot Manager parameters

The Web Robot Manager module does not have a configuration file associated with it, because initially the system contains no predefined Web robots.

Setting Category Tree Manager parameters

The Category Tree Manager module does not have a configuration file associated with it.

Setting Security Manager parameters

The Security Manager settings are loaded through the *SecurityModule.xml* file.

Table 3-40: Security Manager parameters

Parameters	Default	Description
assignSearchRoleToAuthenticatedUser	false	Allows you to assign a query role to an authenticated user.
enableAudit	false	Allows you to log security events such as login, logout, and actions performed on resources.
httpHandlersEnabled	true	Allows enabling or disabling access to all HTTP handlers in *Container.ID.xml* such as the XML APIs, XSDs, or any debug handler that is configured.
sessionTimeoutMinutes	15	Allows you to configure the session expiry time. The session logs are cleaned up either when you log off or when you are inactive for the configured time.

Configuring MIME types

The list of MIME types that Sybase Search can index is in the *MimeTypeMap.xml* configuration file.

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. By default, common formats such as plain text and HTML are indexable, while executable MIME types are not.

You can add custom MIME types and the appropriate text filter in *DocumentFilters.xml*. See “Implementing document filters for unsupported files” on page 120.

Configuring text tokenizers

When a document is indexed, its individual fields are subject to the analyzing and tokenizing filters that can transform and normalize the data in the fields. For example — removing blank spaces, removing html code, stemming, removing a particular character and replacing it with another. Tokenizers splits-up a stream into a series of tokens.

Sybase Search includes several standard text tokenizers that you can use based on your language requirement.

Table 3-41: Standard text tokenizer types

Name	Description
com.isdduk.text.parsing.StdTextTokenizer	StdTextTokenizer extends from BreakIteratorTextTokenizer, and uses java.text.BreakIterator.getWordInstance() for tokenizing sentences. **Note** StdTextTokenizer is suitable for most western languages.
com.isdduk.text.parsing.PreScanBitrTokenizer	PreScanBitrTokenizer extends StdTextTokenizer providing functions that protect user-defined keywords from being destroyed by StdTextTokenizer. Configure defined keywords in *PreScanBitrTokenizer.properties*.

Sybase Search allows you to configure the text tokenizers by modifying the TextProcessor tag in the *TextModule.default.xml* file. See “Setting text tokenizer parameters” on page 70.

Configuring modules using system parameters

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

Enter numeric parameters using:

- 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 large values in an easy-to-read format. Use these formats to also prevent the “missing zero” problem that can sometimes occur when entering numbers that have many parameters 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 35.

Indexing involves three different processes. The first two occur numerous times during one indexing session. The third process occurs as a maintenance operation.

- 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. The greater the memory allocation, the more efficient the entire indexing process is, because more data can be handled simultaneously.

Table 3-42: General upload parameters

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

Writing data to disk

There are two sets of parameters that affect the point at which buffered data is written to indexes—the rate at which the data is written, and the index settings themselves.

Table 3-43: 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.

Parameter	Default	Description
omniq.indexer.sleepFrequency	20	Indicates the number of 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, 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, metadata index segments cache some of their structure in memory to improve indexing and querying performance.
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. Do not change this default value.
omniq.lexicon.document.useRootChildrenCache	true	If set to true, the document lexicon segments will cache some of their structure in memory 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

Index unification is for maintenance and optimization and can take place only after a document store has been reindexed, which produces new index stripes.

Table 3-44: 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 Sybase Technical Support.

Setting Query parameters

All queries are affected by the query parameters. You can scale document stores up or down using the confidence parameter, and the linking parameters affect all “Find Similar” queries.

Table 3-45: Query parameters

Parameter	Default	Description
omniq.query.termLimit	30	The maximum number of terms in a query. If a query exceeds this number, Sybase Search selects the most important omniq.query.termLimit number of terms from the query.

Parameter	Default	Description
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 omniq.query.confidence and has the effect that a higher confidence value lowers the overall scores, while a lower confidence value 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 overall linking query scores.
omniq.query.training.maxDocs	5	The maximum number of training documents used to train a category; the minimum is one.
omniq.query.training.minTerms	5	The minimum number of terms the category query generator uses when creating a new category query. The lower this value, the more generic the generated queries become.
omniq.query.training.maxTerms	10	The maximum number of terms the category query generator uses when creating a new category query. The higher this value, the more specific the generated queries become.
omniq.query.expansion.maxStrength	10	The maximum query strength, used to allow users to enter integer expansion strength. Example from 0 to 10.
omniq.query.expansion.numLinkingDocs	5	The number of linking documents used as the initial training query to obtain the relevant terms to add to the original query.
omniq.query.expansion.minTerms	5	The minimum additional terms to add to the original query from the linking documents.
omniq.query.expansion.maxTerms	10	The maximum additional terms to add to the original query from the linking documents.

Setting up Metadata Paragraph File

The Metadata Paragraph File (MPF) is where Sybase Search stores metadata and text from indexed files. This data is used to construct result sets and to generate plain-text versions of the indexed documents.

The MPF contains the metadata and body text of a number of indexed documents in a compressed format. The first group of MPFs is created in the 0 (zero) directory located in:

install_location\OmniQ\data\DSM-*UID*\DS-*UID*\MPF\0

Subsequent groups are numbered sequentially beginning with 1.

Configuring MPFs

MPF classes compress all the paragraphs from all documents, favoring those 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 added to a paragraph group, which is compressed and written to disk.
- Each paragraph is compressed individually and written to disk.

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

Paragraphs are not all written together, as it is often necessary to read individual paragraphs from disk. Compressing all the paragraphs together forces to read and decompress all paragraphs to access the sole paragraph required. 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 paragraphs from a document fit exactly within the allocated number of paragraph groups.

Configure paragraph grouping using the MPF parameters shown in Table 3-46. The MPF parameters are defined for all document stores in a container and are set in the main container file *Container.ID.xml*.

Table 3-46: 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 directory.
omniq.index.mpf.foldersPerFolder	50	The number of MPF directories stored per directory.
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. By design, Sybase Search usually slightly exceeds this limit.

Optimizing Sybase Search performance

By default, Sybase Search settings are configured for small to medium numbers of documents. For a multiserver setup, you can adjust the settings for JVM, indexing, and querying to provide better performance.

Java Virtual Machine (JVM) settings

The default maximum JVM heap size is 256MB. When Sybase Search is installed on multiple servers for indexing and querying large number of documents, it requires additional memory. In such cases, set the JVM maximum heap size option to its maximum value to accommodate any memory setting changes.

Table 3-47: JVM options

JVM option	Description
-Xms	Minimum or initial heap size
-Xmx	Maximum heap size

The default setting for the maximum heap size is `-Xmx256m`. To allocate 2GB memory to the Sybase Search container running on a machine that has 4GB of memory, update the relevant container start-up script to `-Xmx2048m`.

In most cases, you can leave the minimum heap size set through the JVM option -Xms unchanged. If you know that your installation requires a lot of memory and if cache sizes are large and the maximum amount of memory has been allocated, then you can increase the minimum or incremental heap size to -Xms64m or -Xms128m.

General indexing settings

You may be able to improve performance by evaluating and modifying some index-related configuration settings. Index settings are *Container.n.xml* located in:

*install_location**OmniQ**config*\

where *n* is the container ID.

Index buffer maximum memory

The index buffer maximum memory setting affects indexing performance. This setting determines how much memory is utilized when indexed documents are buffered. When a buffer reaches the maximum limit, the data is written to the indexes and the buffer is emptied.

A large buffer size ensures that more data is indexed each time, making the disk writing phase more efficient. Reducing the number of times the indexing session has to write to disk improves performance; as appending data involves expensive file seek time. This is important especially when the buffer has a large number of items, say upwards of 100,000 items.

The default setting for the index buffer maximum memory setting is:

```
<SystemProperty name="omniq.index.buffer.maxMemory" value="10 MB"/>
```

Choose a new value for the index buffer maximum memory based on your machine specifications and the total memory allocated to the JVM setting of the container. Increase the value on an incremental basis and observe the effect on the indexing time. For example, try setting the value to 16MB or 32MB.

Note Sybase recommends that you keep the index buffer maximum memory to 16MB or 32MB. This is because setting large values for index buffer maximum memory, for example to 64 MB, involves additional in-memory management of the index buffer size. This slows down the indexing performance.

Partial indexing session

Partial indexing sessions are particularly relevant to database imports, and run much faster than incremental updates. Partial indexing sessions target specific data without having to check whether other existing data has been updated or not. See “Indexing document stores” on page 35.

Store indexed text

You can improve indexing and querying performance by not storing indexed text. For indexing, the performance gain is achieved since you are not required to write compressed paragraphs to disk. With querying, the performance gain is achieved by not having to read compressed paragraphs. You can specify whether to store indexed text when creating a document store. See “Document stores” on page 20.

Index sleep time

Data-writing from buffer to disk during indexing is I/O-intensive. Any other applications using the same server compete for disk access time. Sybase Search includes two parameters — index sleep frequency and duration — disabling the sleep duration parameter improves performance.

Index sleep frequency determines how many terms of data are written to disk before an index writing pause. Sleep duration determines the length of this pause, in milliseconds. The default settings are:

```
<SystemProperty name="omniq.indexer.sleepDurationMillis" value="20"/>
<SystemProperty name="omniq.indexer.sleepFrequency" value="100"/>
```

To improve indexing performance, set these values to 0 (zero) to disable index sleeping.

Document store index settings

Index settings for the document stores are in *Container.n.xml*, which is available at:

*install_location**Omniq**config*\

where *n* is the container ID.

The document store index settings affect the indexes created within each *index stripe* for every document store.

Main indexes

If the documents indexed in a container are spread across multiple document stores, changing the main index settings produces lesser performance gain compared to having all the documents in one document store.

Note omniq.index.term.useRootChildrenCache and omniq.lexicon.document.useRootChildrenCache are set to true. Disabling these degrades performance.

Number of term index segment settings

omniq.index.term.numSegments index parameter has a greater impact on query performance, but will have lesser effect on indexing performance. The default setting is:

```
<SystemProperty name="omniq.index.term.numSegments" value="5"/>
```

The main index data is striped across each segment to reduce the Java RandomAccessFile seek time. For larger files, RandomAccessFile seek time gets progressively longer, so by having multiple segments, the overall index size is kept the same, but is split over multiple files. As file lengths are shorter, the seek times in each of the files gets reduced. Increasing the number of index segments increases the number of files handled, which impacts the JVM settings of the container.

By default, the number of term index segments is set to 5. If you intend to index more than 250,000 documents in a document store, Sybase recommends increasing the number of term segments to, for example, 10.

Note For higher numbers of documents (above 500,000), setting the number of index segments to a lower value can cause indexing errors. Indexing more than 2 million documents in a single document store requires 20 term index segments or more.

Number of reverse documents lexicon segment settings

The second main index setting that can affect performance is omniq.lexicon.reverseDocument.numSegments. The default setting is:

```
<SystemProperty name="omniq.lexicon.reverseDocument.numSegments" value="4"/>
```

Increasing the number of reverse document lexicon segments slightly improves performance since it shares the data across more segments. The number of reverse document lexicon segment value must be a factor of the document maximum key length. For the default maximum document key length of 256, different document path lengths are split across 4 segments with lengths 0-64, 64-128, 128-196, and 196-256.

Note The number of reverse documents lexicon segment settings applies only to file system document stores.

Document lexicon maximum key length settings

omniq.lexicon.document.maxKeyLength is the last main index setting parameters that has an impact on Sybase Search performance. The default setting is:

```
<SystemProperty name="omniq.lexicon.document.maxKeyLength" value="256"/>
```

If the source documents have document paths that are less than the default value, reducing maxKeyLength improves performance. When maxKeyLength is reduced, index lexicon files require less space to store the indexed document paths; because there is less data to read, access time is reduced.

Note The number of reverse documents lexicon segment settings applies only to file system document stores.

Metadata indexes

Metadata indexes are similar to main term indexes, but store lesser data compared to main term index data. Typically, all the main term indexes settings work well for metadata indexes settings. The default setting is:

```
<SystemProperty name="omniq.index.metadata.numSegments" value="2"/>
```

If you are indexing more than 250,000 documents in a document store, increase the number of metadata index segments to, for example, 4 or 5.

Metadata Paragraph File (MPF) indexes

MPF indexes store compressed text paragraphs from the indexed documents, which are stored separately from the main calculation data. Unlike the main term and metadata indexes, MPF files are not open all the time, which ensures that the total number of file handles used by the container's JVM is not exceeded.

Documents per folder settings

These settings control how many documents are stored per MPF file, how many MPF files are stored per folder, and the maximum number of folders per folder:

```
<SystemProperty name="omniq.index.mpf.docsPerFile" value="20"/>
<SystemProperty name="omniq.index.mpf.filesPerFolder" value="250"/>
<SystemProperty name="omniq.index.mpf.foldersPerFolder" value="50"/>
```

In most cases, the filesPerfolder and foldersPerFolder does not impact indexing or querying performance, unless Sybase Search is running on an operating system where there is a performance limitation with the distribution of files and folders.

docsPerFile parameter does affect your performance. When you index small documents, the default setting generates smaller MPF files. If you increase docsPerFile to 100, for example, the number of MPF files being generated would reduce by a factor of 5 and ensure more efficient paragraph storage.

If you mostly index large documents, the default setting generates large MPF files, which can lead to long file seek times when retrieving paragraphs. Decrease docsPerFile to 15 or 10, to reduce the average MPF size and reduce seek times.

Paragraph group settings

These settings determine how many paragraphs are grouped together for reading and writing and the maximum number of group entries allowed:

```
<SystemProperty name="omniq.index.mpf.maxParagraphGroups" value="5"/>
<SystemProperty name="omniq.index.mpf.maxTotalGroupEntries" value="50"/>
```

For this particular setting, a higher value for maxParagraphGroups helps in compression since more paragraphs compress better than trying to compress them separately. Depending on your requirement for minimum number of paragraphs for grouping, you should change the default value.

Note The first paragraph returned for a given relevant document may not necessarily be the first paragraph stored in the MPF file. The most relevant paragraph from the first few paragraphs is returned first. Setting maxParagraphGroups to 1 does not mean that only one paragraph is read from the MPF files, in other words, it does not retrieve one paragraph per page.

Index unification settings

Index unification combines multiple index stripes into a single stripe and ensures that the underlying indexes are kept efficient with data.

Index unification reads data for each item from the various stripes, combines the data and writes it to a new index stripe. As with standard document indexing, a buffer stores the data before it is written to disk. Unlike standard document indexing, data is read sequentially, so increasing the unifier buffer settings provides a lesser amount of performance improvement.

For example, item “java” is only read once, unlike the main document indexing, where more data for “java” could be obtained from additional documents and combined with the existing buffer's data for “java”.

Increase these buffer sizes for a slightly better speed unification:

```
<SystemProperty name="omniq.unifier.termMapSizeSoftLimit" value="40K"/>
<SystemProperty name="omniq.unifier.termMapSizeInBytesSoftLimit" value="32MB"/>
<SystemProperty name="omniq.unifier.metadataMapSizeSoftLimit" value="40K"/>
<SystemProperty name="omniq.unifier.metadataMapSizeInBytesSoftLimit" value="32MB"/>
```

These parameters have a larger impact on index unification performance:

```
<SystemProperty name="omniq.unifier.sleepDurationMillis" value="20"/>
<SystemProperty name="omniq.unifier.sleepFrequency" value="100"/>
```

Sleep frequency and duration parameters in index unification work in the same way as for main document indexing. Setting the two parameter value to 0 (zero) effectively disables index sleeping for index unification process.

Note If you have I/O-intensive tasks running simultaneously with index unification, consider increasing the sleep time parameters to ensure that indexing runs more as a background process.

Term Lexicon Module cache settings

The Term Lexicon Module settings and Term Lexicon Module Delegate settings are configured in the *TermLexiconModule.default.xml* and *TermLexiconModuleDelegate.default.xml* available at:

install_location\OmniQ\config

The Term Lexicon Module is used by the hub container and the container from a single server installation. The Term Lexicon Module Delegate is used by any satellite container.

Cache capacity settings

Both modules have a cache for storing the mapping between indexed terms and their internal term IDs. Since character-based terms are too large to be used in indexes and calculations, each term is assigned a unique term ID. The caches in the Term Lexicon Modules allow Sybase Search to obtain the term ID without having to access the Term Lexicon's disk structure. The more terms that a satellite container can cache, the fewer network calls are made to the hub container for unknown terms, improving performance for both indexing and querying.

By default, the cache.capacity setting in the Term Lexicon's configuration file is set to 131072. The cache is most efficient when its value is a power of 2. Since many lexicons generated from a large number of documents can exceed 300,000 terms, consider increasing the cache.capacity to either 262144 or 524288. Increasing the cache on the hub container or single server container improves query time, whereas increasing the cache on a satellite container improves indexing time.

Query performance settings

There are several changes you can make to the Query Module file to improve query-related performance. The file is at:

install_location\OmniQ\config\QueryModule.default.xml

Term statistics cache settings

The term statistics cache holds some calculation data that has been obtained from across the Sybase Search distributed network. If a past query has cached term statistics data for the term “java”, for example, subsequent queries can use the information in the cache, and can avoid having to make additional network calls. Increasing the size of the cache ensures that more data is cached, minimizing the number of network and reducing overall query time.

By default, the cache.termStats.capacity setting in the Query Module’s configuration file is set to 131072. The cache is most efficient when its value is a power of 2. Since large sets of indexed documents can exceed 300,000 terms, you may want to consider increasing cache.termStats.capacity to either 262144 or 524288.

Note This term statistics cache is located on the hub container and every Document Store Manager has one term statistics cache.

Query runner pool size settings

The size of the query runner pool determines how many queries can be processed simultaneously.

By default, the queryRunnerPool.size parameter in the Query Module’s configuration file is set to 20. You can increase the setting to 30, 40, or 50, but there is a natural limit to the number of queries that can be processed simultaneously. The limit depends on the machine specification and other performance tuning settings for each satellite container.

Query data cache settings

The query data cache allows data from previous searches to be cached, reducing the amount of disk access time for subsequent queries that use similar terms.

For example, a user searches for java virtual machine; the data for each individual term is cached. When another user searches for java message queue, only the data for message and queue needs to be obtained from the indexes on the disk as the data for java is in the cache.

Each document store manager has its own query data cache, which is shared by all the document stores managed by that document store manager.

Use the Web administration page for each document store manager to manage the size of each query data cache. By default, each cache is set to 50MB. To improve query time increase the value, if the maximum JVM heap size permits, to 128MB, 256MB, or 512MB.

Query term limit

The query term limit ensures that queries that have more than a given number of terms are reduced, to avoid running long query calculations. By default, the query term limit is set to 30 terms.

```
<SystemProperty name="omniq.query.termLimit" value="30"/>
```

To increase query performance, set to a lower value, for example 20, 15, 10, or even 5. By reducing the query term limit, you can reduce the maximum amount of data being processed for each query, however doing so reduces the accuracy of a query that has more than the maximum terms allowed.

Note A query limit lower than 10 will probably have a noticeable effect on the query results for longer queries.

Performance tuning

There are some general rules that you can apply to your Sybase Search installation for optimum performance.

Indexing

Indexing is more efficient when done in large batches. Large batches allow more data to be combined in memory before it is written to disk.

Improving performance on a deployed network

Networks have two primary characteristics to consider: *bandwidth* and *latency*. If Sybase Search is installed on multiple servers, reducing the effect of any network lag improves the performance of indexing.

For example, closer the source documents are to the container that does the indexing, better are the performance results. If source documents are on a different server than where indexing takes place, each document must be copied across the network for analyzing and processing. If possible, place all documents on the same server to remove any network effect.

Similarly, for database imports, the faster the database connection to the satellite container, the better your performance is, since the speed of the JDBC call to the database depends on network speed.

PDF document filter

The open source document filter, which is used for indexing PDF files can be very slow if you have a large number of PDF files. Alternatively, you can try the Content Adapter filters to see the effect on indexing time. If indexing performance is critical, Content Adapter which is a separately-sold option, may provide the extra indexing speed you require.

Monitoring memory usage

Sybase recommends that you monitor the total JVM memory being used, after you have changes to default configuration values.

From the system administration Web page, click Memory Usage. The Memory Usage page displays the current memory usage by the JVM and other caches.

For example, the Memory Usage page may show that each document store's active stripe is using a lot of memory. If you have many document stores, you may want combine the data into fewer document stores. See “Document stores” on page 20.

Configuring authentication systems

Sybase Search uses Sybase Common Security Infrastructure (CSI) for Java bindings. You can configure these authentication systems:

- Built-in authentication
- LDAP-based authentication

Sybase CSI also supports authentication by other providers.

Built-in authentication

Sybase Search includes basic authentication and authorization functionality.

Table 3-48: Default user accounts and access levels

Account	Password	Role	Access level
sysearch_guest	Blank	SySearch_Guest	Query functionality
sysearch_admin	Set during installation	SySearch_Admin	All functionalites

To use the built-in authentication system, use the *csi.xml* located in: *install_location\OmniQ\config\security\csi\sample\builtin*

```
<?xml version="1.0" encoding="UTF-8"?>
<config:configuration xmlns:config="http://www.sybase.com/csi/1.0/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.sybase.com/csi/1.0/config
http://www.sybase.com/csi/1.0/config.xsd">
      <provider name="com.isdduk.framework.security.provider.BasicAttibuter"
       type="attributer"/>
      <provider name="com.isdduk.framework.security.provider.ResourceCheckAuthorizer"
       type="authorizer" controlFlag="sufficient"/>
      <authenticationProvidername="com.isdduk.framework.security.provider.
       BuiltinLoginModule" type="loginModule" controlFlag="sufficient">
      </authenticationProvider>
</config:configuration>
```

LDAP-based authentication

If the default built-in authentication system does not suit your security requirements Sybase Search also supports LDAP for authentication and authorization. To configure Sybase Search for LDAP, edit *csi.xml* using any text editor and change the authenticationProvider parameters to your LDAP server configurations. The file is located in: *install_location\OmniQ\config\security\csi\sample\ldap*

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration xmlns:config="http://www.sybase.com/csi/1.0/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.sybase.com/csi/1.0/config
http://www.sybase.com/csi/1.0/config.xsd">
       <provider name="com.isdduk.framework.security.provider.ResourceCheckAuthorizer"
       type="authorizer" controlFlag="sufficient"/>
       <provider name="com.isdduk.framework.security.provider.BasicAttibuter"
       type="attributer"/>
       <authenticationProvider name="com.sybase.security.ldap.LDAPLoginModule"
        controlFlag="sufficient">
          <options name="ServerType" value="sunone5"/>
```

```
          <options name="ProviderURL" value="ldap://localhost:389"/>
          <options name="DefaultSearchBase" value="dc=sybase,dc=com"/>
          <options name="BindDN" value="uid=jsmith,dc=sybase,dc=com"/>
          <options name="BindPassword" value="test"/>
       </authenticationProvider>
       <provider name="com.sybase.security.ldap.LDAPAttributer" type="attributer"/>
</configuration>
```

CHAPTER 4

Configuring Web Administration

This chapter describes the key configuration parameters for the Hyena servlet container, 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 default Hyena configuration takes place during installation of the Web administration component of Sybase Search. You can change the configuration of the Hyena servlet container using a text editor to edit the Hyena configuration file called *server.xml*, located in *install_location\Hyena\config*.

Table 4-1: HTTP server tag attributes

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 4-2 shows the attributes for the Request-Handler tag.

Table 4-2: Request-Handler tag attributes

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 4-3 shows the attributes for the Request-Parser tag.

Table 4-3: Request-Parser tag attributes

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 value 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 value are denied.

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

Table 4-4: Request-Keep-Alive tag attributes

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 4-5 shows the attributes for the Remote-Admin tag.

Table 4-5: Remote-Admin tag attributes

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

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

Table 4-6 shows the attributes for the Logging tag.

Table 4-6: Logging tag attributes

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, the time of each HTTP request is written to the log.

Table 4-7 shows the attributes for the Container tag.

Table 4-7: Container tag attributes

Attribute	Default value	Description
debug	false	Prints debugging information to the standard output stream. Information includes the context name and path, the Java libraries used, and details of each servlet that is loaded.

Table 4-8 shows the attributes for the JSP-Handler tag.

Table 4-8: JSP-Handler tag attributes

Attribute	Default value	Description
vigilance	0	The number of seconds that must elapse before the last modified date of the *.JSP* file is checked to determine whether it needs to be recompiled. Set this value to zero or a negative number to never check the file. If this attribute is set to a positive value, you must load the Java compiler library. For more information about obtaining and loading a Java compiler library, contact Sybase Technical Support.

Table 4-9 shows the attributes for the Error-Template tag.

Table 4-9: Error-Template tag attributes

Attribute	Default value	Description
path	*install_location**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 4-10 shows the attributes for the Context tag.

Table 4-10: Context tag attributes

Attribute	Default value	Description
name	None	All context resource URIs implicitly start with this value, which must begin with a forward slash. For example, the context named */omniq* can have its home page at */omniq/index.html*.
path	None	The full path of the directory that contains the context.

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.

Using Sybase Search Web service

Sybase Search Web service uses the Web Services Description Language (WSDL). The WSDL descriptor describes the messages, ports, SOAP bindings, and services that define how communication is carried out between a client application and a service. Sybase Search receives Simple Object Access Protocol (SOAP) messages, which are parsed and translated by the Web service and then transmitted back as SOAP messages to client applications.

Web service message operations

Table 4-11 describes the operations supported by the Sybase Search Web service.

Table 4-11: Web service operations

Name	Description
Query	Allows you to search indexed documents. This operation accepts query parameters and returns results.
GetCategories	Allows you to get a list of categories from a Sybase Search server. This operation does not accept any parameters and returns all the current categories.
GetDocumentGroups	Allows you to get all the documents from a Sybase Search server. This operation does not accept any parameters, and returns all the current document groups.
GetMetadata	Allows you to get all the metadata from a Sybase Search server. This operation does not accept any parameters, and returns all the current metadata groups.
GetRealDocument	Allows you to get a real (binary) document from a Sybase Search server. This operation accepts a document ID, and returns the binary content (encoded by base64) of the original document.
GetDocumentText	Allows you to get the plain text extracted from a document This operation accepts a document ID, and returns the text content of the original document. On the server side, returned text content is first encoded by UTF-8, and later encoded by base64.

Working with attachments

The GetRealDocument and GetDocumentText operations return SOAP messages that contain base64 binary contents. By default, Sybase Search optimizes the transmission of SOAP messages by conforming to the Message Transmission Optimization Mechanism (MTOM) specifications. It creates a separate attachment for base64 encoded, binary data and references it from the main SOAP message.

If you use a Web services middleware that does not support MTOM, returned SOAP messages produce error messages. You can disable MTOM in the configuration files, but doing so results in performance degradation during transmission of SOAP messages.

Setting Web service security and deployment

You can deploy the Web service:

- Into Sybase Search container (default deployment) or,
- Into another J2EE-complaint application server.

Sybase Search provides search features with no access restrictions. In the Web Administration server, you can log in as guest user to access all search features.

You must follow these security policies:

- If the deployment is into the Sybase Search container, the Web service must be accessible to everyone who has access to the container through HTTP.
- If the deployment is into another J2EE-complaint application server, the Web service logs in to the Sybase Search container using guest user credentials. The login process is transparent and the Web service is accessible to users who have access to the application server through HTTP.

CHAPTER 5

Customizing Sybase Search

This chapter contains information about developing, configuring, and using custom HTTP handlers, filters, metadata parsers, and text tokenizers.

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 includes five HTTP handlers, four XML handlers, and a generic file serving handler (for the XML Schema Definitions). You can also develop and plug in custom HTTP handlers as necessary.

See “Configuring the container XML file” on page 61.

XML document groups HTTP handler

This handler returns a list of document groups in an XML format that is compliant with its XML Schema Definition (XSD), available in *install_location/OmniQ/config/xsd/DocumentGroups.xsd*.

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 its document store members for display and integration purposes.

In a default installation of Sybase Search, the XML document groups HTTP handler and its XSD handler are available in:

- *http://<container-host>:<container-port>/xml/documentgroups*
- *http://<container-host>:<container-port>/xsd/documentgroups*

XML metadata HTTP handler

This handler returns a list of all indexable metadata fields in an XML format that is compliant with its XSD, available in *install_location/OmniQ/config/xsd/Metadata.xsd*.

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, the XML metadata HTTP handler and its XSD handler are available in:

- *http://<container-host>:<container-port>/xml/metadata*
- *http://<container-host>:<container-port>/xsd/metadata*

XML query HTTP handler

This handler takes query parameters over HTTP (GET or POST) and returns a result set in an XML format that is compliant with the result set XSD, available in *install_location/OmniQ/config/xsd/ResultSet.xsd*.

In a default installation of Sybase Search, the XML query HTTP handler and its XSD handler are available in:

- *http://<container-host>:<container-port>/xml/query*
- *http://<container-host>:<container-port>/xsd/resultset*

Table 5-1: XML query HTTP handler parameters

Parameter	Description
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: <*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, that a document must achieve to be included in a result set.

Parameter	Description
numParas	The number of document excerpts to return for each document returned in the page of results.
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 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). Adjusting this value 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.
documentStoreAddrs	A comma-delimited list of document store addresses, which are of the form <DSM-ID>-<DS-ID>, where DSM-ID is the document store manager ID and DS-ID is the document store ID. The document store address enables you to specify the document store (or stores) for search query.
localStats	When set to true, the query performed in each document store only uses calculation data obtained from that document store. Cached global statistics are ignored, which changes the relevancy scores of the query results. If you use localStats with documentStoreAddrs, the query runs as an isolated query only on the stores specified in documentStoreAddrs.

XML document HTTP handler

This handler returns the text from an indexed document in an XML format that is compliant with its XSD, available in *install_location/OmniQ/config/xsd/Document.xsd*

In a default installation of Sybase Search, the XML document HTTP handler and its XSD handler are available in:

- *http://<container-host>:<container-port>/xml/document*
- *http://<container-host>:<container-port>/xsd/document*

Table 5-2: 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 that is compliant with its XSD, available in *install_location/OmniQ/config/xsd/Categories.xsd*.

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

In a default installation of Sybase Search, the XML categories HTTP handler and its XSD handler are available in:

- *http://<container-host>:<container-port>/xml/categories*
- *http://<container-host>:<container-port>/xsd/categories*

Configuring HTTP handler security

Enable the HTTP handlers by setting the security flag to true:

1 Using a text editor, open the *SecurityModule.xml* available in *install_location\OmniQ\config*.

2 Set the boolean property named httpHandlersEnabled to true. For example:

```
<Property name="httpHandlersEnabled">
<Value class="java.lang.Boolean">true</Value>
</Property>
```

Note If the value is not set to true, by default HTTP handlers are disabled.

Developing and configuring customized parsers

You can create custom DATE, INT, and FLOAT metadata parsers.

All metadata parsers implement this common base interface:

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

Regardless of whether the convenience base class is used, each custom parser class must provide a no-arguments constructor and implement the appropriate one of these four 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.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

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.TermParser
     parse(java.lang.String source, com.isdduk.util.set.StringSet result :
           boolean
     parse(java.lang.String source) : com.isdduk.util.set.StringSet
     format(java.lang.String term) : java.lang.String
```

The parse method that returns a Boolean result, should contain the parsing logic; the other parse method should simply create a suitable *set* object and delegate the call, because 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 new metadata parsers

After you have compiled the new parser class, 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 *install_location\OmniQ\lib* directories.

Note Place the JAR file into every container's library directory.

See "Metadata parsers" on page 49.

Developing and configuring custom text tokenizers

All values for document body text and textual metadata (excluding file paths) are passed through the configured text tokenizers to be broken into individual terms. Each term that is not preserved, not a stopword, and is neither too short nor too long, is passed to the configured term stemmer to be reduced to its root form. Both the text tokenizers and term stemmer can be reimplemented and reconfigured where necessary. See:

- "Preserved terms" on page 59
- "Stopwords" on page 58

Text tokenizing converts extracted plain text into words. Term stemming reduces words to their common roots. Text tokenizing 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 text tokenizer 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 tokenizer class com.isdduk.text.parsing.StdTextTokenizer 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 (for example, Chinese and Japanese characters) 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, Sybase recommends that you write your own custom text tokenizer. Term splitting algorithms designed for a single language should out-perform the Java BreakIterator, which is designed to handle multiple languages.

Developing custom text tokenizers

Text tokenizers are implemented in pairs, consisting of a non-stateful and a stateful implementation. The non-stateful tokenizer must define the tokenizing algorithm, and the stateful tokenizer must manage a “tokenizing state” (for example, when tokenizing a series of contiguous character buffers). The tokenizers are defined in the following interfaces:

- com.isdduk.text.parsing.TextTokenizer
- com.isdduk.text.parsing.StatefulTextTokenizer

Sybase Search provides abstract, base classes for each, to simplify the implementation and integration of new tokenizer classes:

- com.isdduk.text.parsing.AbstractTextTokenizer
- com.isdduk.text.parsing.AbstractStatefulTextTokenizer

Sybase Search also provides additional support for two common tokenization techniques, which are:

- tokenizing strings using a java.util.BreakIterator
- tokenizing strings using into an array of strings (java.lang.String[])

For each of these techniques, there are additional sub-interfaces and sub-classes:

- com.isdduk.text.parsing.BreakIteratorTextTokenizer
- com.isdduk.text.parsing.BitrStatefulTextTokenizer
- com.isdduk.text.parsing.StringArrayTextTokenizer
- com.isdduk.text.parsing.StrAryStatefulTextTokenizer

This section provides information for developers about developing customized text tokenizers based on your search requirements and language specification.

Extending from BreakIteratorTextTokenizer

Extend BreakIteratorTextTokenizer and realize this method:

```
newBreakIterator(): java.text.BreakIterator
```

This example demonstrates how to use the default com.isdduk.text.parsing.StdTextTokenizer:

```
import java.text.BreakIterator;
public class StdTextTokenizer extends
BreakIteratorTextTokenizer {
…
public BreakIterator newBreakIterator(){
          return BreakIterator.getWordInstance();
}
…
}
```

Extending from StringArrayTextTokenizer

Extend StringArrayTextTokenizer and realize these methods:

- `toWordArray(java.lang.CharSequence, int, int): java.lang.String[]`
- `toWordArray(CharSequence text): java.lang.String[]`

This example demonstrates the expected output for the sample input string.

`"I am John Smith"` returns:

```
Word[0] = "I"
Word[1] = " "
Word[2] = "am"
Word[3] = " "
Word[4] = "John"
Word[5] = " "
Word[6] = "Smith"
Word[7] = "."
```

For more information about the interface, see Javadocs available in: *install_location\webapp\docs\api\index.html*

Configuring the term stemmer

The term stemmer interface 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 to 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 tokenizer and term stemmer

After you have compiled the new text tokenizer class, 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 *install_location\OmniQ\lib* directories.

Note Place the JAR file into every container's library directory.

The Text Manager module loads the text tokenizer and term stemmer. Using a text editor, edit *TextModule.default.xml* in *install_location\OmniQ\config* to change the *text.parsing.xml* property.

Note Perform the metadata changes for the container instance that loads the Text Manager.

Developing custom document filters

You can develop custom document filters for single file formats like PDF, as well as for documents that may contain multiple formats like XML, ZIP, or e-mail messages. The multidocument file support allows separate documents in a single file to be indexed separately. Implement the following interface for the custom document filter:

```
interface DocumentFilter {
     void setName(java.lang.String name);
     java.lang.String getName();
     void setMimeTypes(java.lang.String[] mimeTypes);
     java.lang.String[] getMimeTypes();
     java.util.Properties getProperties();
void setProperties(java.util.Properties);
FilteredDocumentIterator filter(FilterInput fltInput)
            throws java.io.IOException:
}
```

The filter method allows custom document filters to return multiple documents from a single file or input stream argument. You can also implement the following interfaces in *DocumentFilters.xml* for better configuration support:

- com.omniq.flt.InputMatchingFilter
- com.omniq.flt.TempFileFilter

- com.omniq.flt.TimeoutFilter
- com.omniq.flt.TextDocumentFilter

For a detailed information about implementing this interface, see the Javadocs in: *install_location\webapp\docs\api\index.html*

Implementing document filters for unsupported files

This section describes how to implement a custom document filter for file types that are not even supported by Sybase Search Content Adapter.

For a file system document store or a passive (Web) document store, implement your custom document filter by:

1 Adding the MIME type and extension of the new document file type in the *MimeTypeMap.xml* file.

2 Passing the new MIME type and extension to the correct Stellent filter in the *DocumentFilters.xml* file.

For a database document store, the default content-type detector is not capable of recognizing new document file types. For recognizing the new document file type, you can either:

- Use the DOC_CONTENT_TYPE alias in your SQL query, which bypasses the content-type detector. See “Constructing an import SQL statement” on page 26.

  or

- Implement a new Java com.isdduk.io.ContentTypeDetector class and configure this class for using the DB-import SQL query statement. This example illustrates how to implement the new class:

  - The DB-import SQL statement checks the Java system properties to check if the default implementation has been overwritten. Open the *Container.ID.xml* file and include the following Java system property.

    ```
    <SystemProperty name="omniq.import.db.detector"
    value="com.mycompany.MyContentTypeDetector" />
    <SystemProperty
    name="omniq.import.db.detectorReadLimit"
    value="512" />
    ```

    where com.mycompany.MyContentTypeDetector is the name of your new content-type detector class.

The detectorReadLimit is the number of bytes the content-type detector receives from each file, in order for it to determine the content type of the byte data.

Note Create a JAR file for the new class and copy the class in *install_location\OmniQ\lib.*

Configuring for XML content indexing

You can configure the XML content indexing using these options:

- Create a custom XML document filter to handle your XML format.
- Create an XSL style sheet that transforms your XML format into the Sybase Search import XML format.
- Use the default XML document filter.
- Repurpose your XML data into the import XML format such that XML document filter handles XML content automatically.

❖ **Devoloping a custom XSL style sheet**

1 Identify the XML content for indexing. This example indexes the *BookReviews.xml* file having *id*, *isbn*, *author*, *title*, and *reviewer* searchable as metadata, and *reviewtext* searchable as text.

```
<BookReviews>
      <Review id="44399">
     <Book isbn="1401302718">
        <Title>The Rest of Her Life</Title>
        <Author>Moriarty, Laura</Author>
        <Price>$24.95</Price>
     </Book>
     <Reviewer>Castellitto, Linda</Reviewer>
         <ReviewText>
       Like her 2005 debut novel ...
     </ReviewText>
  </Review>
      <Review id="11200">
     <Book isbn="1400063566">
        <Title>Away</Title>
        <Author>Bloom, Amy</Author>
        <Price>$23.95</Price>
    </Book>
```

```
    <Reviewer>McKanic, Arlene</Reviewer>
    <ReviewText>
        Lillian Leyb, the heroine ...
    </ReviewText>
  </Review>
</BookReviews>
```

2 Develop a custom XSL style sheet. This example reformats *BookReviews.xml* file into Sybase Search import XML format:

```
<xsl:template match="BookReviews">
    <Import xmlns="urn:schemas-sybase-com/sysearch-import">
         <xsl:for-each select="Review">
            <Document>
               <Metadata>
                   <Entry>
                      <Name>id</Name>
                      <Value><xsl:value-of select="@id"/></Value>
                   </Entry>
                   <Entry>
                       <Name>isbn</Name>
                       <Value><xsl:value-of select="Book/@isbn"/></Value>
                   </Entry>
                   <Entry>
                       <Name>title</Name>
                       <Value><xsl:value-of select="Book/Title"/></Value>
                   </Entry>
                   <Entry>
                      <Name>author</Name>
                      <Value><xsl:value-of select="Book/Author"/></Value>
                   </Entry>
                   <Entry>
                       <Name>price</Name>
                       <Value><xsl:value-of select="Book/Price"/></Value>
                    </Entry>
                     <Entry>
                       <Name>reviewer</Name>
                       <Value><xsl:value-of select="Reviewer"/></Value>
                     </Entry>
               </Metadata>
               <Content>
                   <xsl:value-of select="ReviewText"/>
               </Content>
            </Document>
</xsl:for-each>
```

3 Edit the *DocumentFilters.xml* file located in *install_location\OmniQ\config\DocumentFilters.xml*. Add a new entry under the document filter class com.omniq.flt.xml.XMLInputMatchingFilter. The purpose of com.omniq.flt.xml.RootElementMatcher class is to inform the root class to apply transformation action to XML files with a root element of BookReviews.

```
<InputMatcher class="com.omniq.flt.xml.RootElementMatcher">
   <Pattern>BookReviews</Pattern>
   <Action type="transform">
         <Params>
               <Param>
                     <Name>StylesheetURL</Name>
                     <Value>file:/D:\XML\transform\BookReviews.xsl</Value>
                </Param>
          </Params>
     </Action>
</InputMatcher>
```

This process transforms the XML content into a format that Sybase Search ImportXML filter can read and index.

Customizing externally managed document stores

There are two important components that are crucial for importing documents into an externally managed document store manager.

- Import client – a client-side object, which is used to send information to an external managed document store manager.
- Import handler – a server-side object for accepting import client requests, handling each request, and returning an appropriate response back to the client.

The import client is implemented as a concrete class: com.omniq.repository.indexing.DocumentIndexingSession

The object defines a set of methods that enables you to authenticate and perform updates on a target external managed document store.

The import handler is an HTTP handler, and is configured in the Sybase Search container XML file. Every external managed document store uses a single import handler. The default URL for the HTTP handler is:

`http://hostname:port/em/indexer`

where:

- *hostname* is the name or IP address of the machine hosting Sybase Search container on which the target external managed document store manager resides.
- *port* is the port number of the Sybase Search container on which the target external maneged document store manager resides.

Note The externally managed (or passive) document store managers require HTTP handlers to be enabled. If the HTTP handlers are disabled on a container, which hosts an externally managed document store manager then it will be unable to start any indexing sessions. Also, the Web robot requires an active external managed document store manager to index its Web pages.

Table 5-3 describes the custom HTTP headers used for client and server communication.

Table 5-3: HTTP import format

Header name	Description
X-Import-Command	The following options are supported: • Begin – starts a new indexing session on the target external managed document store. • Import – posts a new or updated document to the external managed document store. • Remove – requests a document to be removed from the external managed document store. • Abort – exits the indexing session. • End – ends the indexing session. • Ping – notifies the server that the client is still active. • Tell – notifies the server after the client has ended the indexing session; the server notifies the client whether the changes to the indexes are complete.
X-Import-Address	Specifies the address of the document store on which to begin an indexing session. This command is used in conjunction with the Begin command.

Header name	Description
X-Import-Authenticate	Specifies either a user name and password pair, or a security session ID for the server to authenticate. If a user name and password pair has been set explicitly, the authentication token is in the format username:password, where both the user name and password are UTF-8 and base64 encoded. If the user name and password pair has not been set, there must be a thread-local security session ID available for use, in which case the authentication token is a long integer value. This command is used in conjunction with the Begin command. **Note** If neither a user name and password pair or a security session ID is available, an IllegalStateException is thrown.
X-Import-Session-ID	Specifies the ID of the indexing session. The server creates the ID token and returns it by wrapping it with the response to the Begin command. The client stores the value and returns it with every other request it makes to the server. 123-10010_fewk8fba, where the format is made up of the document store address and an unique identifier.
X-Import-Doc-Ref	Specifies the reference of the document which the client is attempting to import or remove.
X-Import-Response	Specifies the server's response to the client request. This is an integer value which maps to an error code, where zero equals success.

This sample code illustrates how third-party applications can import documents into an externally managed document store using the import API.

```
URL url = new URL("http://localhost:7701/em/indexer");
IndexingSessionConfig cfg = new IndexingSessionConfig(url);
DocumentIndexingSession session = new DocumentIndexingSession(cfg);
session.setAuthDetails("robin", "sherwood".toCharArray());
IndexingActionResult result = session.begin("122-10010");
if (result.isSuccessful()) {
      String docRef = "1";
      FastMap metadata = // acquire metadata
      Reader content = // acquire content
      result = session.indexDocument(docRef, metadata, content);
      if (result.isSuccessful()) {
             System.out.println("Document indexed OK");
             session.end();
             while ( ! session.awaitCompletion(1).isSuccessful())
              ;
              System.out.println("Document is now live");
      }
     // else handle error
}
// else handle error
```

CHAPTER 6

Using Sybase Search

This chapter describes how to access Sybase Search and search across documents.

Accessing Sybase Search

You can log in to Sybase Search from any computer that runs a Web browser.

Note The following procedure describes how to access Sybase Search for searching across documents. To perform Sybase Search administrative tasks, you must log in to the Sybase Search administration pages. See "Accessing administration pages" on page 15.

❖ **Log in to Sybase Search**

1 Open a Web browser.

2 In the address bar of the Web browser:

`http://`*`hostname`*`:`*`port`*`/omniq/`

where:

- *hostname* is the name or IP address of the machine hosting the Sybase Search Web application.
- *port* is the port number for the J2EE application server hosting the Sybase Search Web application. The default port number is 8111.

3 If you have administrative privileges enter your user name and password and click Login. If you are a guest user, click the Guest Search link.

Searching across documents

From the Home page, click the Search tab. You can search across all of the documents that have been indexed in Sybase Search. There are various options you can use to get accurate search results:

- From the Search page, enter appropriate values for the following fields and click Search:
  - **Specifying search terms** – specify terms to be included or excluded from the 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.
    - 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 it contains.

      For example, a search for “operating systems” with Not Terms “Windows XP” does not discount a document for containing the phrase “opens in a new window.”
  - **Selecting categories** – categories are a set of documents grouped by content, independent of location or type of document store. You can use categories to filter search results. You can also view lists of documents for each category. See “Categorizing documents” on page 41.
  - **Selecting document groups** – limit your search to one or more predefined document groups by selecting specific groups from the Document Groups list.

- **Specifying 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.

  Select an operator for each metadata parameter. All metadata types support the equal to (=) operator. 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. Values for text types are processed as search text. Numeric type values are processed as numbers, and date type values must be in the configured format, for example, dd/mm/yyyy. See “Metadata parsers” on page 49.

  Note Synonyms and acronyms are not applied to TEXT metadata fields.

Table 6-1: Predefined metadata parameters

Parameter name	Type	Reliable
Author	TEXT	No
Character Count	INT	No
Client	TEXT	No
Comment	TEXT	No
Company	TEXT	No
Creation Date	TEXT	No
Description	TEXT	No
Document Name	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
URL	TEXT	No
Word Count	INT	No

Metadata Combination Operators – you can select two combination operators:

- Within Expression – use 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 authored by only John Smith or John Roberts.

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

  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 You cannot perform a metadata search for multiple Across Expressions operators.

Although you can use predefined metadata parameters with Search Term and Not Term to refine a search, you can also use them independently for a metadata search. Search results from a pure metadata search have no meaningful relevance scores.

- **Setting result options** – you can further refine search results by defining values for these options, located on Result Options tab:
  - Minimum Document Relevance – define the minimum relevance ranking that a document must score for it to be included within the search results.
  - Number of Results per Page – define the number of document results to display for each page.

- Number of Paragraphs per Document – define the number of document paragraphs to display for each result document.
- Score Unknown Terms – when this option is selected, terms unknown to the system (and therefore, do not exist in any indexed document) are considered by the scoring algorithm.
- Term Highlighting – when selected, terms from the query are highlighted in the result paragraphs and in the plain-text versions of the matching document slices, as shown by the view text links.
- Query Expansion Strength – 0 indicates no query expansion performed on the search results; 10 indicates the maximum query expansion performed on the search results.

Understanding search results

Sybase Search provides you with various options that help you understand and use search results effectively.

Paragraph rating stars

Display results with visual indicators that indicate how relevant a result paragraph is to your query. Paragraph rating stars are added to all result pages like main search, view category, linking, and training.

A three-star rating is assigned to a good match, and no star is assigned to a poor match. The results page displays only the most matching paragraph from the source document using your query text. The stars provide visual indicator to the most relevant areas within a document that match your query.

Term highlighting

View highlighted terms for both user-specified and internal queries. For user-specified queries, terms from the query text are highlighted in the search results. Internal queries, which are generated automatically from source documents, for example, using “Find Similar” or “Train Category” queries that highlight the most relevant terms from source documents. See “Categories” on page 42.

For example, perform a Find Similar “Character Stream” query, on a Java document, the internal query result highlights terms like “stream”, “PrintStream”, “Writer”. The highlighted terms are not part of the original query, but have been extracted from the source document as the most relevant words. Similarly, for a Train Category query, “connection.html”, on a Java document about database connection, the result highlights “JDBC”, “connection”, and “URL”; these are the internal terms that generate the category.

APPENDIX A

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 a unique module ID.

Module files

Table A-1: Module generated file locations

Module	File location
Category module	*install_location*\OmniQ\data*CategoryModule-uid* where *uid* is the unique module ID.
Category Tree module	*install_location*\OmniQ\data*CategoryTreeModule-uid* where *uid* is the unique module ID.
Document Group Manager	*install_location*\OmniQ\data*DGM-uid* where *uid* is the unique module ID.
Document Filter module	*install_location*\OmniQ\data*DocumentFilters-uid* where *uid* is the unique module ID.
Document Stores	*install_location*\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.
Document Store Manager	*install_location*\OmniQ\data*DSM-uid* where *uid* is the unique ID.
Metadata Manager	*install_location*\OmniQ\data*MetadataModule-uid* where *uid* is the unique module ID.
Metadata Manager Delegate	*install_location*\OmniQ\data*MetadataModuleDelegate-uid* where *uid* is the unique ID.
Query Manager	*install_location*\OmniQ\data*QueryModule-uid* where *uid* is the unique ID.

Module	File location
Repository module	*install_location*\OmniQ\data*RepositoryModule-uid* where *uid* is the unique ID.
Security module	*install_location*\OmniQ\data*SecurityModule-uid* where *uid* is the unique ID.
Term Lexicon Manager	*install_location*\OmniQ\data*TermLexiconModule-uid* where *uid* is the unique ID.
Term Lexicon Manager Delegate	*install_location*\OmniQ\data*TermLexiconModuleDelegate-uid* where *uid* is the unique ID.
Text Manager	*install_location*\OmniQ\data*TextModule-uid* where *uid* is the unique ID.
Unique ID generator	*install_location*\OmniQ\data*UID-uid* where *uid* is the unique ID.
Web Robot Manager	*install_location*\OmniQ\data*WRM-uid* where *uid* is the unique ID.

APPENDIX B

Sybase Search Content Adapter

This appendix describes how to install and configure the Sybase Search Content Adapter for use with Sybase Data Integration Suite.

Introduction

Sybase Search Content Adapter includes the third-party Stellent document filter that enables searching across proprietary file formats such as Microsoft Office documents and Adobe PDF.

Earlier versions of Sybase Search, which was part of Sybase Data Integration Suite (DI Suite) 1.0, included Stellent document filters. In DI Suite 1.1, Stellent filters and Sybase Search are packaged separately.

You must purchase and install the Sybase Search Content Adapter separately. To install, follow the instructions in “Installation” on page 139.

Note If you are using earlier versions of Sybase Search, you must obtain the license for Sybase Search Content Adapter to use it with Sybase Search 4.0. See “License information” on page 138.

License information

Sybase Search Content Adapter uses the Sybase Software Asset Management (SySAM) licensing mechanism for license administration and asset management. After you have purchased the Sybase Search Content Adapter, go to the SPDC Web site at http://sybase.subscribenet.com to generate and download the license. For complete information about SySAM, see the Sybase Software Asset Management Users Guide.

Sybase Search Content Adapter supports the same license models as the DI Suite Sybase Search. Therefore, to install Content Adapter you must generate the license similar to that generated for Sybase Search. For example, if Sybase Search uses the unserved license model, you must generate unserved licenses for Content Adapter.

For more information on license models that DI Suite supports, see Chapter 1, "Before You Begin," in the *Sybase Data Integration Suite 1.2 Installation Guide*.

License installation steps

The license installation steps are different for served and unserved license model.

❖ **Installing with a served license model**

1 Copy the license file to the *licenses* directory on the network license server machine.

2 Refresh or restart the license server.

❖ **Installing with an unserved license model**

- Copy the license file to the *install_location\SYSAM-2_0\licenses* directory.

Note By default, the SySAM *licenses* directory is located in *install_location\SYSAM-2_0*. If the container on which you want to install the Content Adapter is configured with a different SySAM *licenses* directory, you can find the directory location by looking up the *sysam.license.dir* system property in the *install_location/Search-4_0/OmniQ/config/Container.ID.xml* file, where *ID* is the container ID on which you want to install the Content Adapter.

❖ **Modifying e-mail notifications**

1 From the Environment page, click License Information.

2 Click Modify in the Container Email Notification table.

3 Make the changes and click Modify.

Installation

This section provides instructions for installing the Sybase Search Content Adapter. The Sybase Search Content Adapter installer installs the Content Adapter files in the *Search-4_0* directory.

Preinstallation tasks

Before you install Sybase Search Content Adapter verify that:

- Sybase Search is installed and the *Search-4_0* directory exists on the installation machine.
- SySAM licenses are available in your installation environment. See “License installation steps” on page 138.
- The container on which you want to install Content Adapter is stopped. See “Starting and stopping Sybase Search” on page 12.

Installation mode

Install Sybase Search Content Adapter in one of the following modes:

- GUI mode – the Sybase Search Content Adapter is installed through a graphical user interface. This is the default and recommended, installation mode.
- Console mode – the Sybase Search Content Adapter is installed through a command line interface.

❖ **Installing Sybase Search Content Adapter in GUI mode**

1 Insert the Sybase Search Content Adapter installation media.

- On Windows, the setup program should start automatically. If it does not, start the program manually by selecting Start | Run. Browse to *sysearch_ca.exe*.

- On AIX, Solaris, Linux, and HP, go to the directory where *sysearch_ca.bin* is available and at the command prompt enter:

```
./sysearch_ca.bin
```

2 The Welcome window appears. Click Next.

3 The Readme window displays information about the platforms supported by Sybase Search Content Adapter, related documentation, and technical support. Read the information, and click Next.

4 In License Agreement window, select I accept the terms of the license agreement option and click Next.

5 Verify the installation directory and click Next.

6 Review the information in the Installation Summary window and click Install.

7 When the installation process finishes, click Finish to exit.

❖ **Installing the Content Adapter in console mode**

If no graphics display device is available, or to run the installer without the GUI, use console mode. The flow of the installation is identical to a regular GUI installation, except that the display is written to a terminal window and responses are entered using the keyboard.

1 On Windows, enter:

```
sysearch_ca.exe -console
```

2 On AIX, Solaris, Linux, and HP, enter:

```
./sysearch_ca.bin -console
```

Silent installation

A silent installation (sometimes referred to as an unattended installation) is performed by running the installer and providing a response file that contains answers to all of the installer questions. The response file is a text file that you can edit to change any responses before using it in any subsequent installations.

There are two ways to create a response file:

- Record mode – the installer performs the installation and records all your responses and selections in the specified response file. You must complete the installation to generate a response file.

To create a response file, enter the following command, where *respFileName* is the absolute path of the file name you choose for the response file:

- On Windows:

```
sysearch_ca.exe -options-record respFileName
```

- On AIX, Solaris, Linux, and HP:

```
./sysearch_ca.bin -options-record respFileName
```

You can also use the console mode to generate a response file without using the GUI. To create a response file in console mode, enter the following command, where *respFileName* is the absolute path of the file name you choose for the response file:

- On Windows:

```
sysearch_ca.exe -is:javaconsole -console -
options-record respFileName
```

- On AIX, Solaris, Linux, and HP:

```
./sysearch_ca.bin -is:javaconsole -console -
options-record respFileName
```

This command results in:

- An installation of Content Adapter
- A response file containing all of your responses from the installation

  If you use this response file for a silent installation, the resulting installation is identical to the one from which the response file was created: the same installation location, same feature selection, and all the same remaining information

- Template mode – the installer creates a response file containing commented-out values for all required responses and selections. However, you need not install the Content Adapter, and you can cancel the installation after the response file has been created.

  To create a template file, enter the following command, where *respFileName* is the absolute path of the file name you choose for the response file:

  - On Windows:

```
syssearch_ca.exe -is:javaconsole -options-
template respFileName
```

- On AIX, Solaris, Linux, and HP:

```
./syssearch_ca.bin -is:javaconsole -options-
template respFileName
```

If you run this command in console mode, as shown in the previous example, the installer displays a message indicating that the template creation was successful. In GUI mode, the installer does not display the message.

If you use this response file for a silent installation, the default values for all responses are used. Edit the template with the values you want to use during installation.

Installing interactively using a response file

An interactive installation using a response file allows you to accept the default values from the response file, or to change any of those values for the specific installation. This is useful when you have multiple similar installations that have minor differences that you want to change at installation time.

Enter the following at the command line, where *respFileName* is the absolute path of the response file.

- On Windows:

```
syssearch_ca.exe -options respFileName
```

- On AIX, Solaris, Linux, and HP:

```
./syssearch_ca.bin -options respFileName
```

Installing in silent mode

A silent mode installation allows you to install the Content Adapter with all responses being taken from the response file that you have set up. There is no user interaction. This is useful when you want multiple identical installations, or to automate the installation process.

Enter the following at the command line, where *respFileName* is the absolute path of the response file. The -W option specifies that you agree with the Sybase License Agreement text.

- On Windows:

```
syssearch_ca.exe -is:javaconsole -silent -options respFileName -W
SybaseLicense.agreeToLicense=true
```

- On AIX, Solaris, Linux, and HP:

```
./syssearch_ca.bin -is:javaconsole -silent -options respFileName -W
SybaseLicense.agreeToLicense=true
```

Except for the absence of the GUI screens, all actions of the installer are the same, and the result of an installation in silent mode is exactly the same as one performed in GUI mode with the same responses.

Postinstallation tasks

After installing the Content Adapter, restart the container on which it was installed. See “Starting and stopping Sybase Search” on page 12.

Testing the installation

To test the installation and working of Sybase Search Content Adapter:

- Verify that a directory named *sx* is created under the install *install_location\search-4_0* directory and that this directory is not empty.
- Verify that the Content Adapter license is applied correctly to the container:
  a Log in to Web administration pages. See “Accessing administration pages” on page 15.
  b Select System | Environment. Depending on the installation type, you see one or more containers and the corresponding details.
  c Click License Information on the container on which Content Adapter is installed.
  d Review the license information. If the Content Adapter license is applied correctly, the Content Adapter license information must appear in a separate table for the container to which you applied this license. Also, in this table, the feature name should appear as SYSEARCH_CONTENT_ADAPTER.
- Verify that you can create and index a document store that contains documents that were previously unsupported. See:
  - “Document stores” on page 20
  - “Indexing document stores” on page 35

Uninstallation

Before uninstalling Sybase Search Content Adapter:

- Log out of Sybase Search Web administration pages.
- Stop the container on which Content Adapter is installed. See “Starting and stopping Sybase Search” on page 12.

Uninstalling in GUI mode

❖ **Uninstalling in GUI mode**

1. Invoke the uninstall program:
   - On Windows, select Start | Settings | Control Panel | Add or Remove Programs. Select Sybase Search Content Adapter 1.1 and click Change/Remove.
   - On AIX, Solaris, Linux, and HP, at the command prompt, enter:

```
install_location/Search-4_0/_jvm/bin/java -cp
install_location/Search-4_0/_uninst/uninstall.jar run
```

2. The Welcome window appears. Click Next to continue.
3. The Uninstall window appears, displaying the following features that you can uninstall:
   - Common Container Config – replaces the DocumentFilters and MimeTypeMap settings with the base Sybase Search configuration settings, which were stored in a backup file created during the Content Adapter installation.
   - Content Adapter – uninstalls the Sybase Search Content Adapter.

   Select the feature you want to uninstall and click Next.
4. Review the information in the Uninstallation Summary window and click Uninstall.
5. When the uninstallation process has completed, Click Finish to exit.

Uninstalling in console mode

❖ **Uninstalling in console mode**

1 Invoke the uninstaller:

- On Windows, change to *install_location**sx**_uninst*\ and enter:

```
uninstaller.exe -console
```

- On AIX, Solaris, Linux, and HP, at the command prompt, enter:

```
install_location/Search-4_0/_jvm/bin/java -cp
install_location/Search-4_0/_uninst/uninstall.jar run -console
```

2 Select the feature to uninstall.

Configuring Sybase Search Content Adapter

Sybase Search uses the Content Adapter, which includes Stellent document filter for parsing many document formats. The Stellent document filter is used to extract text from more than one document format—in other words, the same filter instance handles more than one MIME type.

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.

Setting Document Filter parameters for Content Adapter

The Document Filter parameters are loaded through the *DocumentFilters.xml* configuration file. See "Setting Document Filter parameters" on page 73.

The Sybase Search Content Adapter related filters in the configuration file are:

- SearchML export filter – the filter used to parse all configured MIME types. The output from the SearchML export filter is XML-formatted to match the Stellent SearchML XSD that contains the raw text and associated document metadata.
- SearchML filter – an internal filter used to parse the SearchML XML output from the SearchML export filter as given above.

Table B-1: Document Filter parameters for Content Adapter

Parameter	Value
General settings	
timeout millis	45,000
TempFiles keep	false
SearchML filter settings	
className	com.omniq.filter.stellent.SearchMLFilter
SearchML export multi-filter settings	
className	com.omniq.filter.stellent.SearchMLFilterExport
exepath	*install_location/sx/exporter_v2.exe*
fallbackformat	FI_ASCII8
fiflags	SCCUT_FI_NORMAL
xmldefmethod	DTD
embeddingsflag	no
noxmldeclarationflag	no
suppressproperties	no
processgeneratedtext	yes
suppressattachments	yes
suppressarchivesubdocs	yes
bold	no
italic	no
underline	no
doubleunderline	no
outline	no
hidden	no
strikeout	no
smallcaps	no
allcaps	no
originalcharset	no
linespacing	no
lineheight	no
leftindent	no
rightindent	no
firstindent	no
offsettracked	no

Index

D

E

F

G

H

I

Q

R

S

T

U

W

X