[MS-FSO]:
FAST Search System Overview

Abstract

This document describes the intended functionality of the Microsoft® FAST™ Search system and how the protocols within this system interact. It provides examples of some common user scenarios. It does not restate the processing rules and other details that are specific to each protocol. Those details are described in the protocol specifications for each of the protocols and data structures that make up this system.

The FAST Search system is designed to crawl, process, and index content, and to provide search results for that content in response to requests from protocol clients. The system consists of a collection of search-related components, services, protocols, structures, and supporting protocol servers.

Revision Summary

Date	Revision History	Revision Class	Comments
11/06/2009	0.1	Major	Initial Availability
02/19/2010	1.0	Major	Updated and revised the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.05	Minor	Clarified the meaning of the technical content.
11/15/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.06	Editorial	Changed language and formatting in the technical content.
03/18/2011	1.06	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.06	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Microsoft® FAST™ Search system is a server-based system that responds to requests from protocol clients about content that was crawled, processed, and indexed by the system. To respond to those requests, the system maintains and uses data in an inverted index, which is a data structure that contains all of the searchable keywords, numbers, and text that was extracted and processed from items in various content sources.

The following diagram provides a high-level overview of the system.

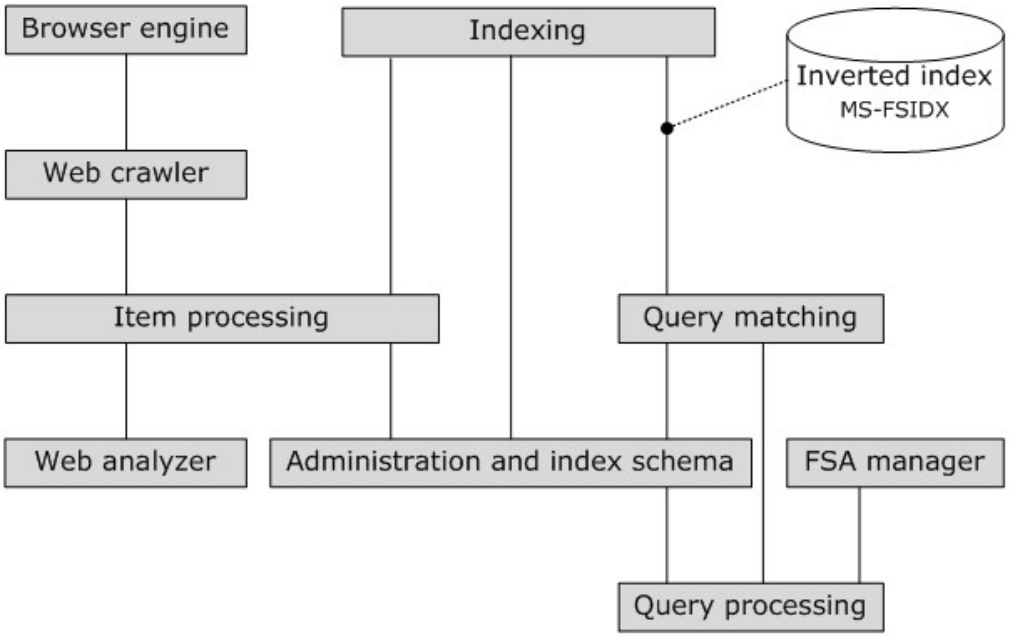

Figure 1: Overview of the FAST Search system

When implementing the system, the preliminary steps are to retrieve data from content sources and submit that data to the item processing service. This can be performed by using an external indexing connector or the internal Web crawler service. An external indexing connector is a software component that is not part of the system but is designed to retrieve and submit content to the item processing service that is provided by the system. The Web crawler service, which is part of the system, is designed to traverse and download content from Web sites. It uses the Web analyzer service to analyze link structures within the content. It also uses the browser engine service to retrieve content from special types of Web pages, such as pages that are generated by script.

After data is retrieved from content sources and submitted to the item processing service, the item processing service extracts and performs linguistic operations on the data. It then sends the resulting data to the indexing service, which in turn creates inverted indexes that are based on that data. The indexing service then sends these inverted indexes to the query matching service for use in query processing operations.

The query processing service performs operations, such as synonym expansion, on queries that are sent by protocol clients and then calls the query matching service. The query matching service uses the inverted indexes to retrieve items that are relevant to and match a query, and then returns those items as a list of query hits. The list of query hits consists of references to the original items,

and the titles and other properties of those items. The query processing service ensures that the list contains references to only those results that the user is authorized to access.

In addition to these services, the FAST Search system offers several administrative services to help manage the system. The index schema service can be used to perform administration tasks that define the search experience for users, such as specifying which properties are visible to users and how property extraction is performed. The administration component offers additional control of the search experience by offering services such as logging, supplying shared storage for configuration files and binary resources, and providing communication middleware. Finally, the FAST Search Authorization (FSA) manager service can be used to grant or restrict user access to items that were processed and added to a search index.

The FAST Search system is designed to run on as few as one computer or virtual server to multiple computers or virtual servers. In the latter case, each computer or virtual server can run one or more system services. Similarly, some services can be distributed across multiple computers or virtual servers. The use of multiple computers or virtual servers enables the system to index more items, perform more updates, and respond to more queries per second than running the system on fewer computers or virtual servers.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Domain Name System (DNS)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Internet Protocol security (IPsec)
Kerberos
NT LAN Manager (NTLM) Authentication Protocol

The following terms are defined in [MS-OFCGLOS]:

abstract object reference (AOR)
administration component
anchor text
best bet
browser engine
Cheetah
claim
client proxy
content distributor
content source
context boost
crawl
crawl site
deep refinement
dictionary
duplicate server
FAST Index Markup Language (FIXML)
FAST middleware
FAST Search Authorization (FSA)
FAST Search Interface Definition Language (FSIDL)
forms authentication
freshness boost
front-end Web server

hit highlighted summary
hyperlink
index column
index schema
indexer row
indexing component
indexing connector
indexing dispatcher
indexing node
indexing service
internal property
inverted index
item identifier
item processing
managed property
multinode scheduler
name server
node scheduler
proximity boost
query hit
query matching component
query matching node
query processing
query processing node
query refinement
rank
rank profile
ranking
search clickthrough
search row
spell tuning
static rank
stemming
token
Web analyzer
Web crawler

1.2 References

We conduct frequent surveys of the informative references to assure their continued availability. If you have any issue with finding an informative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[JDBC] Sun Microsystems, Inc., "The Java Database Connectivity (JDBC) API", JDBC 3.0 API, http://java.sun.com/javase/6/docs/technotes/guides/jdbc/

[MSDN-MIDL] Microsoft Corporation, "Microsoft Interface Definition Language (MIDL)", http://msdn.microsoft.com/en-us/library/ms950375.aspx

[MS-FSADS] Microsoft Corporation, "Administration Database Schema"

[MS-FSAS] Microsoft Corporation, "Administration Services Protocol Specification"

[MS-FSBEPS] Microsoft Corporation, "Browser Engine Processing and Status Protocol Specification"

[MS-FSCADM] Microsoft Corporation, "Crawler Administration and Status Protocol Specification"

[MS-FSCCFG] Microsoft Corporation, "Crawler Configuration File Format Specification"

[MS-FSCDBS] Microsoft Corporation, "Connector Database Schema"

[MS-FSCDCFG] Microsoft Corporation, "Component Distribution Configuration File Format Specification"

[MS-FSCDFT] Microsoft Corporation, "Content Distributor Fault Tolerance Protocol Specification"

[MS-FSCF] Microsoft Corporation, "Content Feeding Protocol Specification"

[MS-FSCHT] Microsoft Corporation, "Cheetah Data Structure"

[MS-FSCMT] Microsoft Corporation, "Crawler Multinode Transport Protocol Specification"

[MS-FSCMW] Microsoft Corporation, "Configuration Middleware Protocol Specification"

[MS-FSCX] Microsoft Corporation, "Configuration (XML-RPC) Protocol Specification"

[MS-FSDP] Microsoft Corporation, "Document Processing Protocol Specification"

[MS-FSDPD] Microsoft Corporation, "Document Processing Distribution Protocol Specification"

[MS-FSDQE] Microsoft Corporation, "Distributed Query Execution Protocol Specification"

[MS-FSFDMW] Microsoft Corporation, "FAST Distributed Make Worker Protocol Specification"

[MS-FSFIXML] Microsoft Corporation, "FIXML Data Structure"

[MS-FSFQL] Microsoft Corporation, "Fast Query Language Structure"

[MS-FSIADM] Microsoft Corporation, "Indexer Administration and Status Protocol Specification"

[MS-FSICFG] Microsoft Corporation, "Indexer Configuration File Format"

[MS-FSID] Microsoft Corporation, "Indexing Distribution Protocol Specification"

[MS-FSIDFT] Microsoft Corporation, "Indexing Dispatcher Fault Tolerance Protocol Specification"

[MS-FSIFT] Microsoft Corporation, "Indexer Fault Tolerance Protocol Specification"

[MS-FSIN] Microsoft Corporation, "Input Normalization Data Structure"

[MS-FSIPA] Microsoft Corporation, "Index Publication and Activation Protocol Specification"

[MS-FSIXDS] Microsoft Corporation, "Index Data Structures"

[MS-FSL] Microsoft Corporation, "Logging Protocol Specification"

[MS-FSLRDS] Microsoft Corporation, "Linguistic Resource Data Structure"

[MS-FSMW] Microsoft Corporation, "Middleware Protocol Specification"

[MS-FSNC] Microsoft Corporation, "Node Controller Protocol Specification"

[MS-FSPSCFG] Microsoft Corporation, "Processor Server Configuration File Format Specification"

[MS-FSQR] Microsoft Corporation, "Query and Result Protocol Specification"

[MS-FSQRC] Microsoft Corporation, "Query and Result Configuration Protocol Specification"

[MS-FSQRCFG] Microsoft Corporation, "Query and Result Configuration File Format Specification"

[MS-FSRFC] Microsoft Corporation, "Remote File Copy Protocol Specification"

[MS-FSRFCO] Microsoft Corporation, "Remote File Copy Orchestration Protocol Specification"

[MS-FSRS] Microsoft Corporation, "Resource Store Protocol Specification"

[MS-FSSAC] Microsoft Corporation, "Search Authorization Connector Protocol Specification"

[MS-FSSACFG] Microsoft Corporation, "Search Authorization Configuration File Format"

[MS-FSSADFF] Microsoft Corporation, "Search Authorization Data File Format"

[MS-FSSADM] Microsoft Corporation, "Search Administration and Status Protocol Specification"

[MS-FSSAS] Microsoft Corporation, "Search Authorization Synchronization Protocol Specification"

[MS-FSSCFG] Microsoft Corporation, "Search Configuration File Format Specification"

[MS-FSSPRADM] Microsoft Corporation, "SPRel Administration and Status Protocol Specification"

[MS-FSSPRDF] Microsoft Corporation, "SPRel Data File Format"

[MS-FSST] Microsoft Corporation, "Spelltuning File Format Specification"

[MS-FSWAADM] Microsoft Corporation, "WebAnalyzer Administration and Status Protocol Specification"

[MS-FSWADF] Microsoft Corporation, "WebAnalyzer Data File Format"

[MS-FSWASDR] Microsoft Corporation, "WebAnalyzer/SPRel Data Receiving Protocol Specification"

[MS-FSWASDS] Microsoft Corporation, "WebAnalyzer/SPRel Data Serving Protocol Specification"

[MS-FSWASMT] Microsoft Corporation, "WebAnalyzer and SPRel Multinode Transport Protocol Specification"

[MS-FSWCU] Microsoft Corporation, "WebAnalyzer/Crawler Utility Structure Specification"

[MS-FSXTAPI] Microsoft Corporation, "XML-RPC Translatable API Structure Specification"

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary"

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary"

[MS-SEARCH] Microsoft Corporation, "Search Protocol Specification"

[RFC2616] Fielding, R., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., and Stewart, L., "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999, http://www.ietf.org/rfc/rfc2617.txt

[XML-RPC] Winer, D., "XML-RPC Specification", June 1999, http://www.xmlrpc.com/spec

2 Functional Architecture

The following sections describe the functional architecture of the Microsoft® FAST™ Search system.

2.1 Overview

The following diagram provides a high-level overview of communications that occur between protocol clients and protocol servers that are part of or used by the Microsoft® FAST™ Search system. It also indicates which protocols are primarily used for those communications.

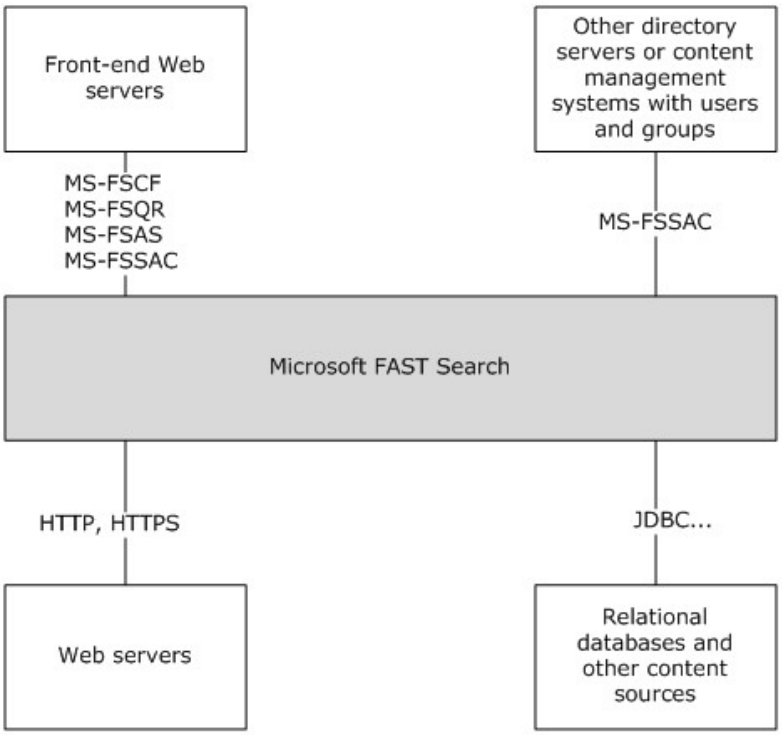

Figure 2: Overview of system communications

As shown in the preceding diagram, the FAST Search system can retrieve items from Web servers, file shares, and other types of **content sources**. It does so by using an **indexing connector**, as described in [MS-FSCF]. The system uses **HTTP** and **HTTPS** to gain access to external Web servers and file shares, and it can authenticate those external sources by using **Kerberos**, the **NT LAN Manager (NTLM) Authentication Protocol**, HTTP authentication, as described in [RFC2617], or **forms authentication** methods. In addition, the system can retrieve items from relational databases by using the Java Database Connectivity (JDBC) database access API, as described in [JDBC]. For more information about how the system retrieves items from relational databases, see [MS-FSCDBS].

The system initially processes each item by using the following procedures:

- Extracting searchable text.
- Detecting the language.

- Retrieving **managed properties** and **internal properties**, such as company names, people names, locations, and dates.

The system indexes and stores the resulting data in an **inverted index**, which enables the system to process and match search queries. A protocol client can be used to customize how items are processed, such as specifying what types of properties to extract.

Front-end Web servers communicate with the system, as described in [MS-FSQR], to enable users to search indexed items and view search results. The system uses an inverted index to return items that match a user query. The items are returned as a list of **query hits** and that list is sorted according to the assumed relevance of each query hit to the specified query. As described in [MS-FSAS], a protocol client can be used to customize how to return the results.

The system is designed to run on a single computer or virtual server, or to scale to a large number of computers or virtual servers. In the latter case, each computer or virtual server can run one or more system services. By duplicating certain services, the system can index a greater number of items or respond to more queries per second.

2.1.1 Services

The following diagram illustrates the primary services of the Microsoft® FAST™ Search system and the relationships between them. It also indicates, by using the abbreviated name of the relevant specification, which protocols and file formats are primarily used by and in communications between those services.

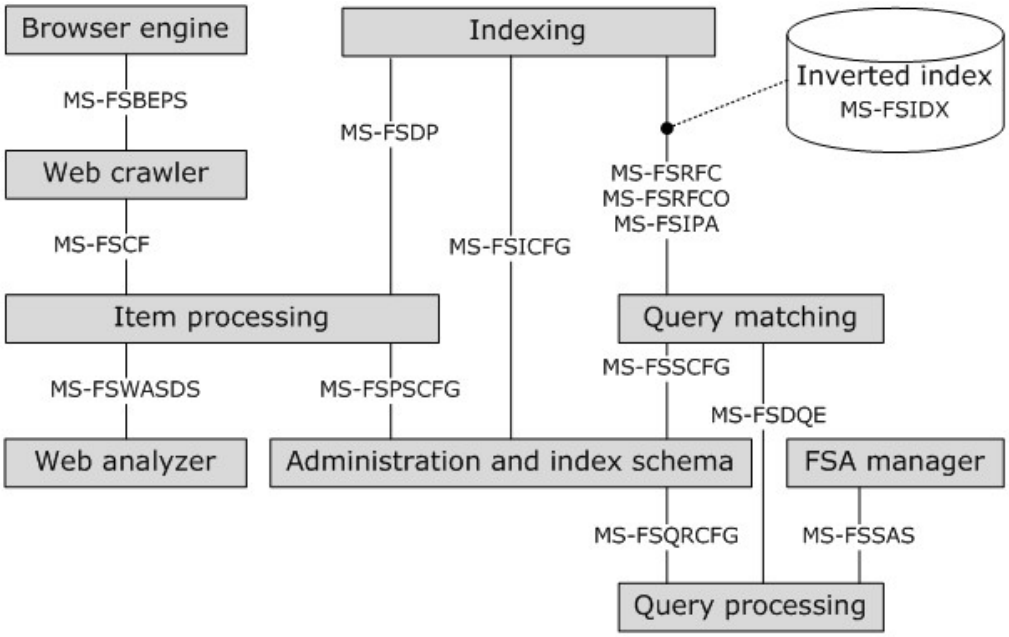

Figure 3: Overview of system services and communications

The following sections describe the functionality and related protocols and file formats for each of these services.

2.1.1.1 Web Crawler Service

The Web crawler service is designed to traverse content sources, retrieve content from those sources, and submit relevant content to the **item processing** service. It uses the **Web analyzer** service to analyze link structures within the content. It uses the **browser engine** service to retrieve content from special types of Web pages, such as pages that generate content by using script.

2.1.1.1.1 Distributed Components and Related Services

The following diagram illustrates the components of the **Web crawler** service, the relationship between the Web crawler service and the browser engine service, and the protocols that define communications between them.

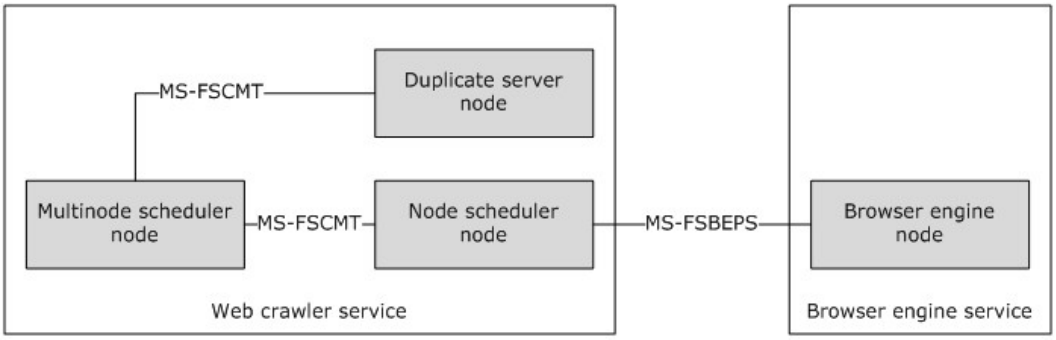

Figure 4: Components of the Web crawler service

The primary components of the Web crawler service are the following:

- **Node scheduler:** Crawls content and submits items to the search index according to configuration settings and a defined schedule.
- **Multinode scheduler:** Manages all of the **node schedulers** for the search index.
- **Duplicate server:** Maintains a database of URIs and content checksums for the Web crawler service. The data is used to duplicate detection across node schedulers in a multinode configuration.

Although it is not part of the Web crawler service, the browser engine service enables the Web crawler service to retrieve data from content that is generated automatically. The Web crawler service obtains this data by sending processing requests to the browser engine service, as described in [MS-FSBEPS].

If the Web crawler service is distributed across multiple computers or virtual servers, all of the communications between the components are defined by the Crawler Multinode Transport Protocol, as described in [MS-FSCMT].

To determine how to perform a crawl, the Web crawler services reads a set of configuration files, as described in [MS-FSCCFG]. Those files specify settings such as which Web servers to **crawl** and the maximum number of pages to download from each Web server. To start, stop, and suspend operations by the various components of the Web crawler service, the service uses the administrative operations that are described in [MS-FSCADM].

2.1.1.1.2 Scalability and Fault Tolerance

To optimize performance and fault tolerance, the Web crawler service can be deployed to multiple computers or virtual servers. The following diagram illustrates how the service can be deployed and the protocols that are used in communications between the components of the Web crawler.

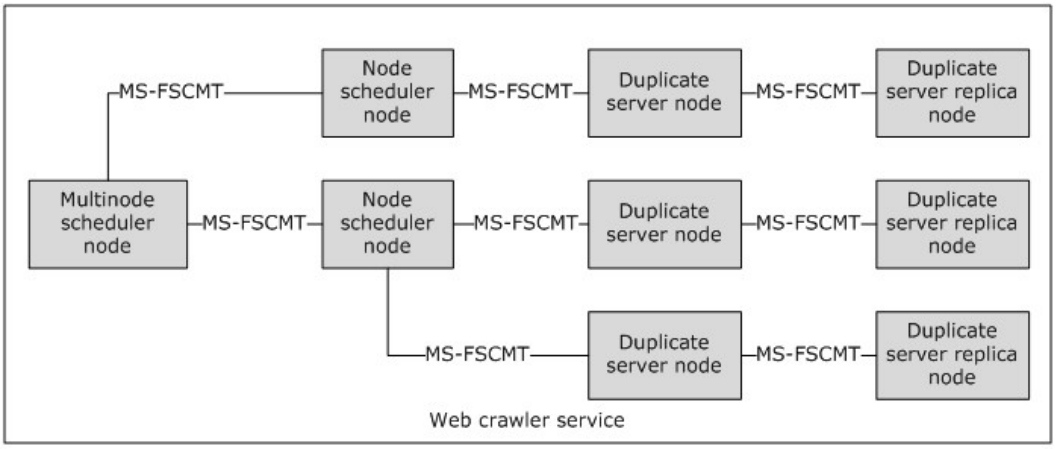

Figure 5: Web crawler service distributed across multiple computers or virtual servers

In this type of deployment, the **multinode scheduler** controls operations of the Web crawler service. It also performs look-up operations against **Domain Name Systems (DNS)** and assigns **crawl sites** to each node scheduler. Each node scheduler crawls the sites that the multinode scheduler assigns to it. The multinode scheduler also communicates with one or more **duplicate servers**, and each duplicate server detects duplicate items across all of the participating node schedulers.

A duplicate server replica serves as a backup for a specific duplicate server and it provides fault tolerance by communicating with that duplicate server. In a multinode configuration, there can be only one duplicate server replica for each duplicate server. For more information, see [MS-FSCMT].

2.1.1.2 Browser Engine Service

The browser engine service processes Web pages by extracting content from those pages, including any links to other Web pages, and downloading and evaluating those pages, including any external dependencies such as external script files and style sheets. The browser engine service returns the resulting HTML and link structures to the Web crawler service.

2.1.1.2.1 Distributed Components and Related Services

The browser engine service receives processing requests from the Web crawler service, as described in [MS-FSBEPS]. It retrieves Web pages for the Web crawler service by using HTTP, as described in [RFC2616].

2.1.1.2.2 Scalability and Fault Tolerance

To maximize the number of requests that can be processed and to provide fault tolerance, the browser engine service can scale from a small to a large number of computers or virtual servers. The following diagram illustrates an implementation in which the browser engine service is distributed across three computers or virtual servers. It also indicates which protocols are used in communications between those computers or virtual servers.

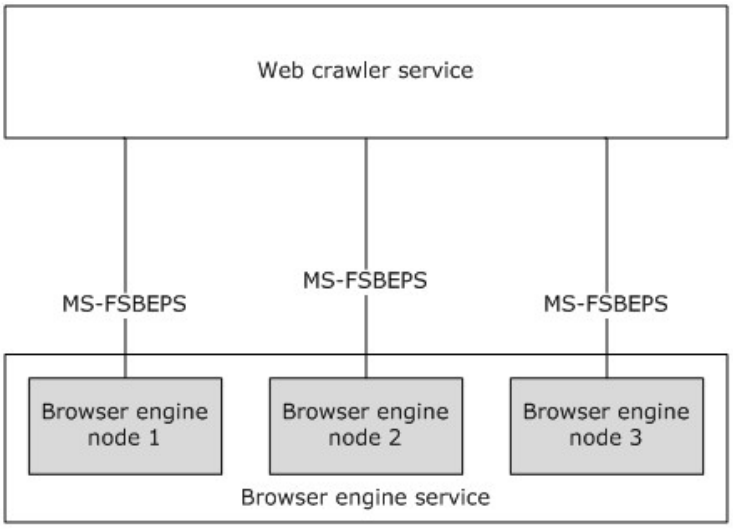

Figure 6: Browser engine service distributed across multiple computers or virtual servers

The Web crawler service is responsible for balancing requests across browser engine nodes, detecting whether a node has stopped working, and, if a node has stopped working, using a different node to provide fault tolerance.

2.1.1.3 Web Analyzer Service

The primary functions of the Web analyzer service are to analyze **search clickthrough** logs and **hyperlink** structures. The analysis produces metadata, such as **rank** scores and **anchor text**, for each item, which contributes to an improved **ranking** of items in search results. Items that show many clicks in a search clickthrough log are deemed popular and therefore receive higher rank scores than items that have fewer clicks. Similarly, items that serve as the destination of many links in many other items are deemed popular and therefore receive higher rank scores than items that serve as the destination of fewer links.

2.1.1.3.1 Distributed Components and Related Services

The Web analyzer service consists of the following types of components:

- **Search clickthrough:** Retrieve search clickthrough logs, which are in binary format, as described in [MS-FSRS], and control the analysis of those logs. In combination with the Web analyzer component, this type of component also splits each analysis job into a series of tasks and distributes those tasks across available processing components, as described in [MS-FSFDMW]. A protocol client is used to manage a search clickthrough component and retrieve status information for it, as described in [MS-FSSPRADM].
- **Web analyzer:** Receive data such as anchor text and URLs from the item processing service, as described in [MS-FSWASDR], and control the analysis of hyperlink structures based on that data. In combination with a search clickthrough component, this type of component also splits each analysis job into a series of tasks and distributes those tasks across available processing components, as described in [MS-FSFDMW]. A protocol client is used to manage a Web analyzer component and retrieve status information for it, as described in [MS-FSWAADM].

- **Processing:** Prepare and process data for analysis by search clickthrough and Web analyzer components, as described in [MS-FSWASMT], and store analysis results in one or more lookup database components.
- **Lookup database:** Store analysis results as key-value pairs. This type of component functions as a database server and it supports use of the WebAnalyzer/SPRel Data Serving Protocol, as described in [MS-FSWASDS], to access the data that it contains.

The Web analyzer and search clickthrough components always run on the same **indexing node**, which is referred to as the Web analyzer node.

To serialize metadata about items and facilitate data transfer, components use the WebAnalyzer/Crawler Utility structure, as described in [MS-FSWCU].

If a system has multiple processing components, the data is transferred as files between those components during the analyses, as described in [MS-FSFDMW], and each file has a data schema, as described in [MS-FSSPRDF] and [MS-FSWADF].

2.1.1.3.2 Scalability and Fault Tolerance

To reduce the total amount of time that is required for an analysis job, the Web analyzer service can be deployed to a large number of computers or virtual servers. The following diagram shows a deployment that contains three lookup database nodes and two item processing nodes. It also indicates which protocols are used in communications between components. For clarity, the diagram does not show all possible communication channels.

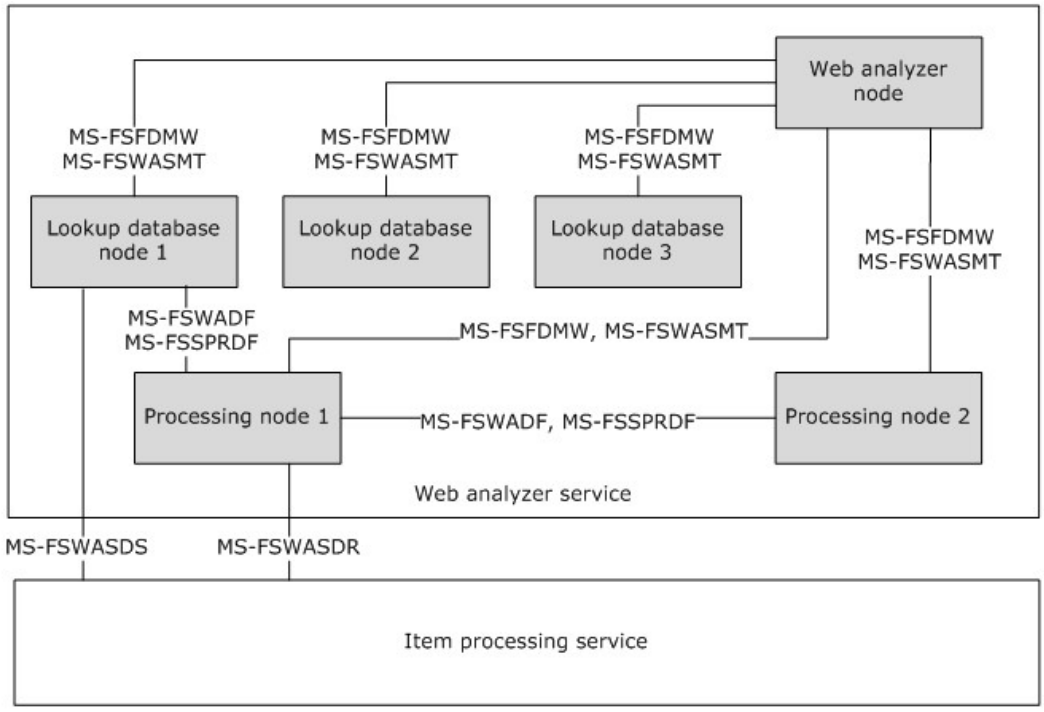

Figure 7: Web analyzer service distributed across multiple computers or virtual servers

The search clickthrough and Web analyzer components always run on the same node, which is referred to as the Web analyzer node. A Web analyzer node cannot be replicated.

2.1.1.4 Item Processing Service

The item processing service manages the end-to-end feeding protocol between indexing connectors, which retrieve items from Web servers and other types of content sources, and the indexing service, which creates inverted indexes based on the items that it receives. The item processing service provides the following services:

- Receives items to be indexed from indexing connectors, as described in [MS-FSCF].
- Extracts content from documents in various formats.
- Discovers and sets managed properties of items.
- Performs linguistic processing on the content of items.
- Sends processed items to the indexing service as batches of data, which optimizes processing performance.

The feeding protocol is asynchronous and uses callbacks to track the status of the item batches that are moving through the feeding chain. The item processing service forwards these status callbacks to the indexing connector that originally submitted the items.

2.1.1.4.1 Distributed Components and Related Services

The primary component of the item processing service is a **content distributor** component that receives items and returns callbacks, as described in [MS-FSCF]. The content distributor component also forwards the items to an available item processing component, as described in [MS-FSDPD]. The item processing component then processes the items and submits them for indexing, as described in [MS-FSDP]. The payload of this transfer is formatted in the **FAST Index Markup Language (FIXML)**, as described in [MS-FSFIXML].

In addition, the item processing service reads a set of configuration files from the configuration component. The configuration files define the following types of processing rules:

- How to normalize characters in items, as described in [MS-FSIN].
- How to configure the item processing nodes, as described in [MS-FSPSCFG].
- How to process items, as described in [MS-FSPSCFG].
- How to use linguistic resources, such as **dictionaries**, as described in [MS-FSLRDS].
- How to use the **spell tuning** component, as described in [MS-FSST].
- Which format to use when submitting items to the indexing service, as described in [MS-FSSCFG].

The item processing service also communicates with the Web analyzer service. It transfers item metadata, such as URLs, hyperlinks, and anchor text, to the Web analyzer service, as described in [MS-FSWASDR]. The Web analyzer service sends updated item metadata, including rank scores, to the item processing service, as described in [MS-FSWASDS].

2.1.1.4.2 Scalability and Fault Tolerance

The item processing service can scale from a small to a large number of computers or virtual servers. Increasing the number of item processing nodes enables the service to process more items in a given amount of time.

The following diagram illustrates how the service can be configured and optimized for performance and fault tolerance, and the protocols that are used in communications between components.

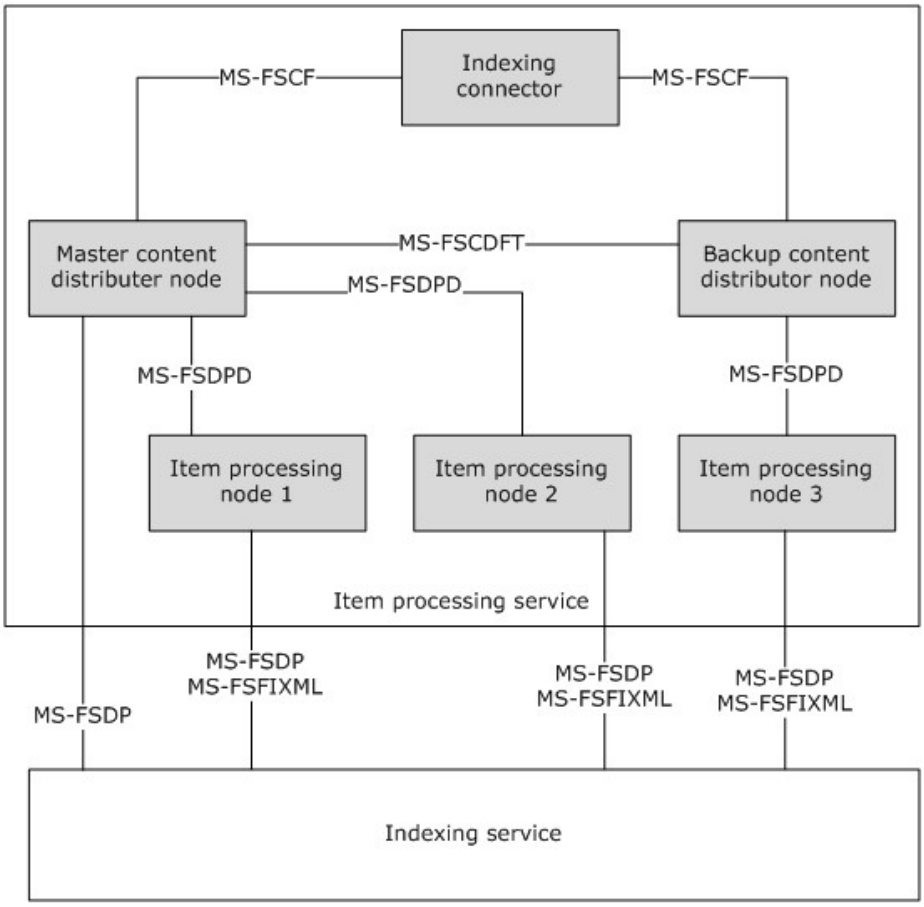

Figure 8: Item processing service distributed across multiple computers or virtual servers

As illustrated in the preceding diagram, the item processing service communicates with all of the content distributor nodes, as described in [MS-FSCF], and communication does not occur between item processing nodes.

Each content distributor node sets up a session for a specific item processing node, as described in [MS-FSDP]. During that session, an indexing connector transfers items to the item processing node, as described in [MS-FSDPD].

The master content distributor node distributes sessions to backup content distributor nodes for item processing and it balances the load across item processing nodes, as described in [MS-FSCDFT]. The master content distributor node communicates with the indexing service, as described in [MS-FSDP], to establish the initial feeding session.

After items are processed, an item processing node submits the items to the indexing service, as described in [MS-FSDP], in the FIXML format, as described in [MS-FSFIXML].

2.1.1.5 Indexing Service

The indexing service creates inverted indexes based on the items that it receives from the item processing service. The indexing service sends these inverted indexes to the query matching service for use during **query processing**, as described in [MS-FSIPA], [MS-FSRFCO], and [MS-FSRFC].

2.1.1.5.1 Inverted Index Structure

An inverted index is a data structure that maps every **token** in a set of indexed items to a list of items that contain the token. It also specifies which item properties can be searched and returned in search results.

Tokens include ordinary language words, numbers, and alphanumeric sequences. During query processing, the query matching service searches for the query in the inverted index. If the inverted index contains the query as a token, the query matching service uses the list of items in the inverted index to return items that match the query.

2.1.1.5.2 Distributed Components and Related Services

The indexing service consists of two components, the **indexing dispatcher** and the **indexing component**. The indexing dispatcher receives items from the item processing service, as described in [MS-FSDP], and then forwards those items to the indexing component, as described in [MS-FSIDFT]. The indexing component then creates or updates the inverted index based on those items.

The following diagram illustrates the relationship between components of the indexing service, the indexing and item processing services, and the protocols that are used in communications between those components and services.

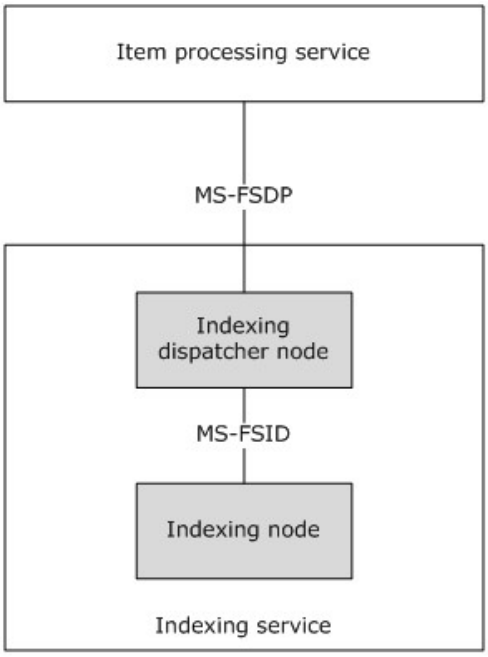

Figure 9: Relationship between components of the indexing service

In addition to the components that are contained within it, the indexing service uses and reads a set of configuration files that are managed by a configuration component. The configuration component

controls both the indexing process, as described in [MS-FSICFG], and the structure of the inverted index, as described in [MS-FSSCFG].

To start, stop, and suspend operations by indexing components, the indexing service uses the administrative operations that are described in [MS-FSIADM].

2.1.1.5.3 Scalability and Fault Tolerance

If the indexing service runs on a single indexing node, both the number of items that it can handle per second and the total number of items that it can add to an inverted index are limited. Therefore, the indexing service is designed to scale to the number of items in the system.

If the indexing service runs on multiple indexing nodes, the indexing dispatcher components and the indexing components are also deployed to multiple indexing nodes. In addition, indexing service components and **query matching components** are typically deployed to different computers or virtual servers to help avoid bottlenecks in the system.

Performance can also be optimized by deploying index data across multiple **index columns**. In this type of deployment, each index column contains only one part of the inverted index, and the combined set of index columns constitutes the entire index. Consequently, each indexing node handles only a subset of the entire index. If an inverted index contains multiple index columns, two or more of the inverted indexes need to be combined to produce consistent search results.

The following diagram illustrates an indexing service that is configured across three index columns, and the protocols that are used in communications between the indexing nodes.

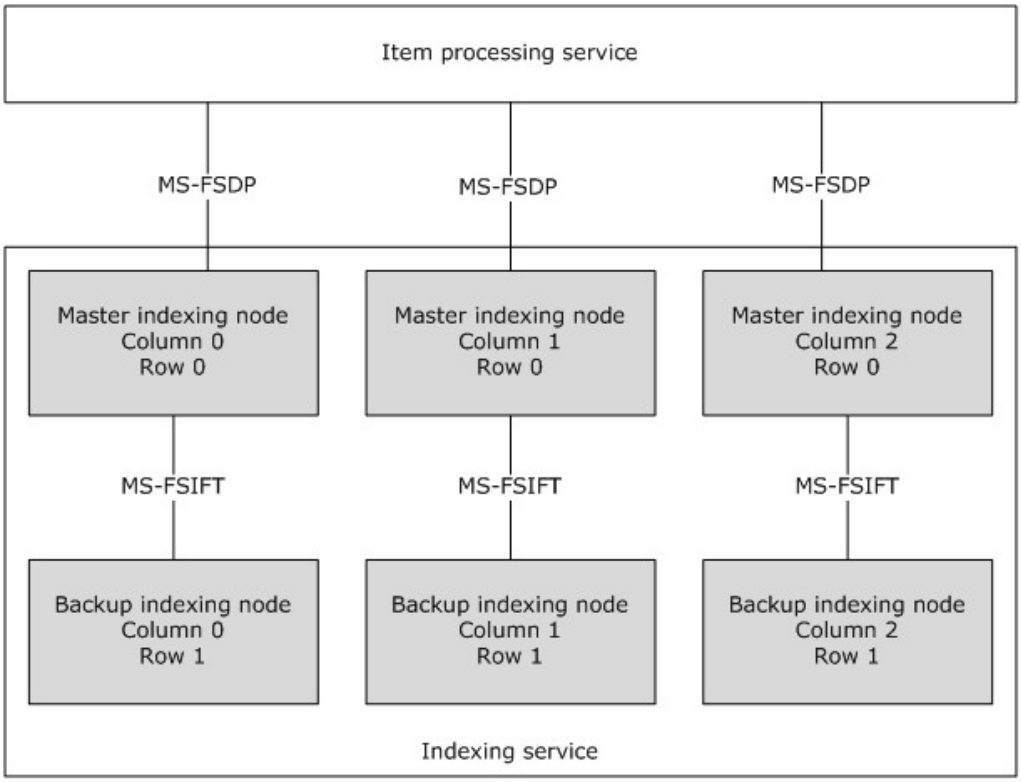

Figure 10: Indexing service configured across three index columns

As illustrated in the preceding diagram, it is also possible to have more than one **indexer row**. This type of configuration provides fault tolerance because the indexing service assigns one master indexing node and one or more backup indexing nodes to every index column. If a master indexing node stops working or becomes unavailable, one of the backup indexing nodes takes over.

It is also possible to use multiple indexing dispatchers for both fault tolerance and performance reasons, as described in [MS-FSIDFT]. In this type of configuration, Row 0 has one master indexing dispatcher running on one of the indexing nodes, and optionally one or more backup indexing dispatchers are running on the other master indexing nodes.

2.1.1.6 Query Matching Service

The query matching service uses inverted indexes, which are created by the indexing service, to retrieve items that match a query and to return those items as a list of query hits. The query matching service receives query requests from the query processing service, as described in [MS-FSDQE].

A query typically contains one or more terms that are combined with query operators, such as AND and OR. In such cases, the query matching service looks up each term in the inverted index and retrieves a list of items in which the term appears. In the case of an AND operator, for example, the list of items consists of the set of items that contain all of the terms. This set of items provides the basis for the list of query hits that the query matching service returns as a response to a query.

The query matching service also looks up the properties of each item, such as the title, author, and time of indexing, and includes those properties in the list of query hits. The order in which the items are returned is based on the requested sorting mechanism, which is usually a complex ranking that is calculated from various item properties or a simple numeric or alphanumeric sort that is based on one or more item properties.

The query matching service is additionally responsible for maintaining aggregation data structures that enable **deep refinement** of query results across large result sets. This type of refinement enables drilling into query results based on aggregated statistical data that was computed for all of the query results.

When generating results, the query matching service can return a **hit highlighted summary** for each item in the list of query hits. A hit highlighted summary contains a fragment of content from the original item and the matching query terms are highlighted in that fragment.

2.1.1.6.1 Distributed Components and Related Services

The query matching service communicates primarily with the following services and components:

- **Indexing service:** Sends inverted indexes individually to the query matching service, as described in [MS-FSRFC], [MS-FSRFCO] and [MS-FSIXDS]. The query matching service then activates the index set as a whole, as described in [MS-FSIPA].
- **Query processing service:** Issues queries and retrieves search results from the query matching service, as described in [MS-FSDQE].
- **Configuration component:** Provides a set of configuration files that are read by the query matching service. Those files specify how to return search results and which item properties are available in the inverted index. For more information, see [MS-FSSCFG].

2.1.1.6.2 Scalability and Fault Tolerance

To optimize performance, the query processing service can alternate between available **search rows** — for example, by using a round-robin algorithm. If a search row does not respond, the query processing service can exclude it from future queries until it becomes available.

The following diagram illustrates a query matching service that consists of six nodes, each of which is configured with three index columns and two search rows. The query processing service chooses a search row, sends an identical query to each column in that search row, and then merges the results that are returned.

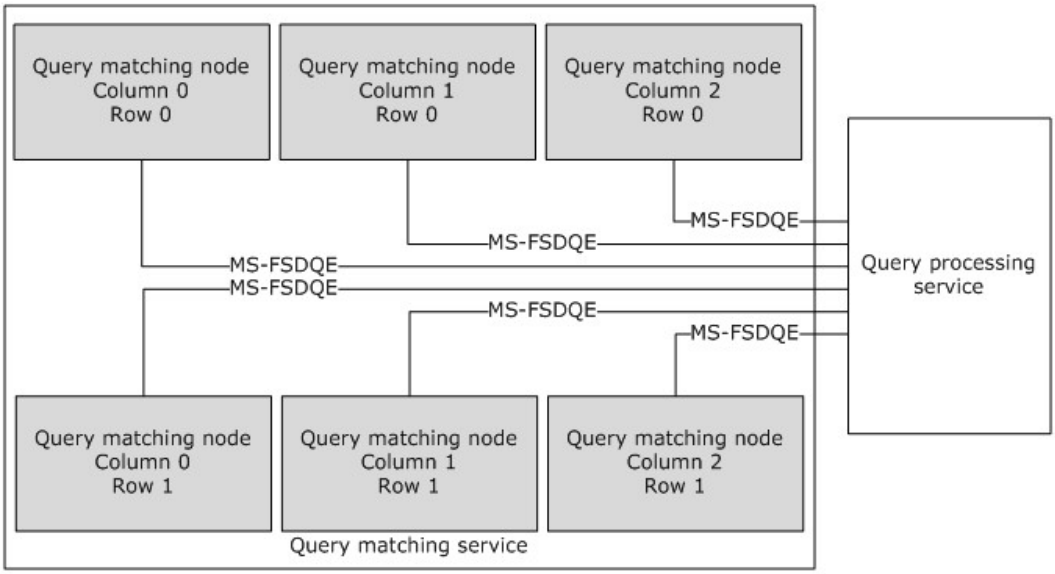

Figure 11: Query matching service configured for three columns and two search rows

The number of index columns in the query matching service always equals the number of index columns in the **indexing service**. Such is the case because the columns represent a partitioning of the index, and each query matching node can handle only one such partition of the index.

If a configuration includes multiple search rows, the search rows are duplicates of each other. The duplicated search rows provide fault tolerance and increase capacity.

2.1.1.7 Query Processing Service

The query processing service performs query and result processing operations that do not require an inverted index. Query-processing operations include parsing the query language, performing linguistic operations, and checking security settings for users and items. Result-processing operations include merging the results from multiple index columns, formatting the list of query hits, formatting **query refinement** data, and removing duplicate items.

The query processing service submits a processed query to the query matching service, as described in [MS-FSDQE]. Each query is sent to one **query matching node** for each search row.

The query processing service is also responsible for ensuring that the user who performs a query receives only those results that the user is authorized to access. The query processing service therefore checks the user's rights and rewrites the incoming query with an access filter that

corresponds to the current user. The **FSA** manager shares the requisite rights information with the query processing service, and the query processing service rewrites the incoming query.

2.1.1.7.1 Distributed Components and Related Services

Various applications send queries and receive results from the query processing service, as described in [MS-FSQR], [MS-SEARCH], and [MS-FSFQL]. The query processing service communicates primarily with the following services and components:

- **Query matching service:** Receives processed queries from the query processing service, as described in [MS-FSDQE].
- **Configuration component:** Provides configuration files, as described in [MS-FSQRCFG] and [MS-FSSCFG], that are read by the query processing service, as described in [MS-FSCX] and [MS-FSCMW]. The query processing service also exposes an interface that enables it to be notified of updates to configuration files, as described in [MS-FSQRC].
- **FSA manager:** Shares authentication and authorization information with the query processing service, as described in [MS-FSSADFF], [MS-FSSACFG], and [MS-FSSAS].

2.1.1.7.2 Scalability and Fault Tolerance

To increase the number of queries that can be handled per second, the query processing service can be deployed across multiple computers or virtual servers. In this type of configuration, all of the nodes need to be configured identically. This means that master query processing nodes and backup query processing nodes do not exist, and communication between such components is therefore not needed. To balance the processing load for incoming queries, the query processing service can also use various hardware- or software-based solutions.

2.1.1.8 Administration Service

The administration service provides common system administration services such as logging, supplying shared storage for configuration files and binary resources, and providing communication middleware. In addition, the administration service includes an **administration component** that can be used to define aspects of the search experience, such as how to perform property extraction, which synonyms to use, and which items are **best bets**.

2.1.1.8.1 Components

The primary components of the administration service are the configuration component, which updates and propagates configurations across computers, and the resource store component, which does the same for binary resources.

2.1.1.8.2 Middleware

The middleware consists of a communication protocol that is used by several services in the system. It facilitates synchronous function calls and asynchronous message callbacks across computer boundaries. The services can exchange both basic data types and information that is represented in the **Cheetah** serialization format, as described in [MS-FSCHT].

The basic data types and the Cheetah format can be described in a high-level manner by using the **FAST Search Interface Definition Language (FSIDL)**, as described in [MS-FSMW]. FSIDL can also be used to describe server interfaces in the middleware. FSIDL specifications are similar to Microsoft Interface Definition Language (MIDL) specifications, as described in [MSDN-MIDL]. The requisite details include how to format FSIDL specifications and map them to remote method

specifications, and how data types in FSIDL specifications map to middleware types. For more information, see [MS-FSMW].

The middleware is based on two roles: protocol client and protocol server. A protocol client initiates communication by using a **client proxy** to call a remote method, including any parameters. The protocol server receives the request from the client proxy, executes the method in an implementation-specific manner, and sends a return value.

To create a client proxy that can be used to communicate with a specific server object, a protocol client requires an **abstract object reference (AOR)**. When a server object is instantiated, a server object URI is constructed from the AOR, and that URI includes information such as the network host name and port for the server object. Similarly, a client proxy creates a server method URI from the AOR when it calls a remote method.

Protocol clients and protocol servers do not explicitly manage the AORs. Instead, the **name server** component associates a logical name with each server object. A protocol client can then contact the name server and request a specific server object, based on the logical name of the object.

2.1.1.8.3 Distributed Components and Related Services

The administration service manages all of the processes in the Microsoft® FAST™ Search system, as described in [MS-FSNC]. It also transports configuration files to and from the configuration component, as described in [MS-FSCMW] and [MS-FSCX], and uses the resource store component to update and propagate binary resources, as described in [MS-FSRS]. The administration service also maintains a main log server component, which collects log messages from local log server components, as described in [MS-FSL]

Several services communicate with the administration service by using **FAST middleware**, as described in [MS-FSMW] and [MS-FSCHT].

2.1.1.8.4 Scalability and Fault Tolerance

Neither scalability nor fault-tolerance mechanisms are provided for the administration service. All of the components need to run on the same computer or virtual server, except the internal database of the administration component. This database can run on a separate computer or virtual server, as described in [MS-FSADS]. The following diagram illustrates this type of configuration.

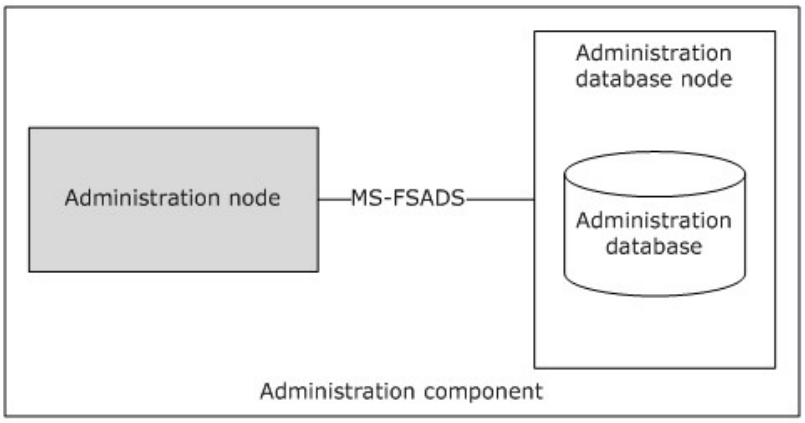

Figure 12: Administration component with the database running on a different computer

2.1.1.9 Index Schema Service

The index schema service manages the **index schema**, which contains all of the configuration entities that are needed to generate configuration files for all of the other services in the system. The index schema service also exposes an API through which protocol clients can view and modify the index schema, as described in [MS-FSAS].

The index schema specifies which and how to index managed properties of items, and which properties can be returned in a list of query hits. The index schema also includes a **rank profile**, which specifies how rank is calculated for each item in a list of query hits. This calculation factors weights for the following components and measures:

- **Static rank**
- **Freshness boost**
- **Proximity boost**
- **Context boost**

2.1.1.9.1 Distributed Components and Related Services

The index schema service receives updates to the index schema from the search administrator and it returns information about the schema to that administrator, as described in [MS-FSAS]. The internal state persists, as described in [MS-FSADS].

Most types of changes to the configuration of the index schema initiate updates to the item processing, indexing, query matching, and query processing services, as described in [MS-FSPSCFG] and [MS-FSSCFG].

2.1.1.9.2 Scalability and Fault Tolerance

Neither scalability nor fault-tolerance mechanisms are provided for the index schema service.

2.1.1.10 FAST Search Authorization Manager Service

The FAST Search Authorization (FSA) manager service receives security updates from indexing connectors and pushes those updates to all of the **query processing nodes** in the system. In addition, the service ensures that security-related aspects of those nodes are configured the same way and it synchronizes such configuration changes across all of the query processing nodes.

2.1.1.10.1 Distributed Components and Related Services

The FSA manager service receives security updates from indexing connectors, as described in [MS-FSSAC]. The FSA manager service then pushes those updates to the query processing service, as described in [MS-FSSAS], [MS-FSSADFF], and [MS-FSSACFG].

2.1.1.10.2 Scalability and Fault Tolerance

Neither scalability nor fault-tolerance mechanisms are provided for the FSA manager service.

2.2 Protocol Summary

The following table provides a comprehensive list of the member protocols of the Microsoft® FAST™ Search system.

Several protocols are based on the XML-RPC protocol, as described in [XML-RPC]. These protocols are specified in a custom format that translates to the XML-RPC protocol. For more information, see [MS-FSXTAPI].

Protocol name	Description	Short name
Administration Database Schema	Enables persistence of the data structures that are used by the administration component and the index schema service.	[MS-FSADS]
Administration Services Protocol	Enables the search administrator to manage synonyms, best bets, and the index schema.	[MS-FSAS]
Browser Engine Processing and Status Protocol	Enables the browser engine service to process and evaluate complex Web pages, and retrieve the resulting output.	[MS-FSBEPS]
Cheetah Data Structure	Enables custom marshaling of data types during communication with FAST middleware.	[MS-FSCHT]
Component Distribution Configuration File Format	Specifies which indexing nodes run which services. The system uses this information during installation and installation-related configuration tasks.	[MS-FSCDCFG]
Configuration (XML-RPC) Protocol	Enables protocol clients to exchange configuration files with protocol servers by using the XML-RPC protocol, as described in [XML-RPC].	[MS-FSCX]
Configuration Middleware Protocol	Enables protocol clients to exchange configuration files with protocol servers by using FAST middleware.	[MS-FSCMW]
Connector Database Schema	Enables persistence of status information for items that are being fed into the system.	[MS-FSCDBS]
Content Distributor Fault Tolerance Protocol	Enables content distributors to determine which distributor is the master content distributor and which distributors are backup content distributors.	[MS-FSCDFT]
Content Feeding Protocol	Enables the feeding of items into the system.	[MS-FSCF]
Crawler Administration and Status Protocol	Enables the search administrator to start, stop, and suspend Web crawler components.	[MS-FSCADM]
Crawler Configuration File Format	Enables the search administrator to configure properties that are related to the Web crawler service.	[MS-FSCCFG]
Crawler Multinode Transport Protocol	Enables synchronization and communication between multiple Web crawler nodes.	[MS-FSCMT]
Distributed Query Execution Protocol	Enables the query processing service to call and pass specific queries to the query matching service, and retrieve the result set.	[MS-FSDQE]
Document Processing Distribution Protocol	Enables communication between components of the item processing service.	[MS-FSDPD]
Document Processing Protocol	Enables the item processing service to transfer items to the indexing service.	[MS-FSDP]
FAST Distributed Make Worker Protocol	Enables the Web analyzer service to control, monitor, and coordinate analyses.	[MS-FSFDMW]

Protocol name	Description	Short name
FAST Query Language Structure	Specifies the format of queries that are processed by the query processing service.	[MS-FSFQL]
FIXML Data Structure	Specifies a format for transferring, storing, and using items in the inverted index.	[MS-FSFIXML]
Index Data Structures	Enables the transfer, storage, and use of data in the inverted index.	[MS-FSIXDS]
Index Publication and Activation Protocol	Enables the indexing service to transfer new index sets to the query matching service and to activate those index sets.	[MS-FSIPA]
Indexer Administration and Status Protocol	Enables administrative control of indexing nodes.	[MS-FSIADM]
Indexer Configuration File Format	Enables the detailed configuration of how indexing is performed, such as which managed properties are searched.	[MS-FSICFG]
Indexer Fault Tolerance Protocol	Enables fault tolerance between multiple indexing nodes.	[MS-FSIFT]
Indexing Dispatcher Fault Tolerance Protocol	Enables fault tolerance and load balancing across multiple indexing dispatchers.	[MS-FSIDFT]
Indexing Distribution Protocol	Enables items to be transported from an indexing dispatcher component to an indexing component.	[MS-FSID]
Input Normalization Data Structure	Enables the normalization of characters in items and queries.	[MS-FSIN]
Linguistic Resource Data Structure	Specifies how the item processing service uses linguistic resources such as dictionaries.	[MS-FSLRDS]
Logging Protocol	Enables remote indexing nodes to send log messages to a central log server on an administration node.	[MS-FSL]
Middleware Protocol	Enables protocol clients to call methods that are located in a different address space.	[MS-FSMW]
Node Controller Protocol	Enables services, components, and file transfers to be controlled and monitored remotely.	[MS-FSNC]
Processor Server Configuration File Format	Enables configuration of the item processing service.	[MS-FSPSCFG]
Query and Result Configuration File Format	Enables configuration of the query processing service.	[MS-FSQRCFG]
Query and Result Configuration Protocol	Enables configuration of properties of the query processing service.	[MS-FSQRC]
Query and Result Protocol	Enables applications to send queries to and retrieve results from the system.	[MS-FSQR]
Remote File Copy Orchestration Protocol	Enables the transfer of files between the indexing service and the query matching service by setting up a file transfer channel.	[MS-FSRFCO]

Protocol name	Description	Short name
Remote File Copy Protocol	Enables the transfer of files between the indexing service and the query matching service.	[MS-FSRFC]
Resource Storage Protocol	Enables services to transfer binary resources to a resource store.	[MS-FSRS]
Search Administration and Status Protocol	Enables administrative control of query matching nodes.	[MS-FSSADM]
Search Authorization Configuration File Format	Specifies the search authorization configuration data that is used to update the query processing service.	[MS-FSSACFG]
Search Authorization Connector Protocol	Enables the FAST Search Authorization (FSA) manager to update the security information that is used during query processing.	[MS-FSSAC]
Search Authorization Data File Format	Specifies the search authorization data that is used to update the query processing service.	[MS-FSSADFF]
Search Authorization Synchronization Protocol	Enables the query processing service to be updated with search authorization data.	[MS-FSSAS]
Search Configuration File Format	Enables the configuration of parameters that are related to the query matching service.	[MS-FSSCFG]
Spelltuning File Format	Enables the transfer of spell tuning data across computer boundaries.	[MS-FSST]
SPRel Administration and Status Protocol	Enables administration of the search clickthrough component of the Web analyzer service.	[MS-FSSPRADM]
SPRel Data File Format	Enables the Web analyzer service to transfer search clickthrough logs between components.	[MS-FSSPRDF]
WebAnalyzer Administration and Status Protocol	Enables management of the Web analyzer service.	[MS-FSWAADM]
WebAnalyzer and SPRel Multinode Transport Protocol	Enables the Web analyzer service to set up analyses and collect analysis results.	[MS-FSWASMT]
WebAnalyzer Data File Format	Enables the transfer of Web pages and link structures that are used during link analyses.	[MS-FSWADF]
WebAnalyzer/Crawler Utility Structure	Enables the marshaling of data types.	[MS-FSWCU]
WebAnalyzer/SPRel Data Receiving Protocol	Enables the transfer of item metadata, such as URLs, hyperlinks, and anchor text.	[MS-FSWASDR]
WebAnalyzer/SPRel Data Serving Protocol	Enables the transfer of item metadata that was updated with rank scores.	[MS-FSWASDS]
XML-RPC Translatable API Structure	Enables a custom format to be translated to the XML-RPC protocol, as described in [XML-RPC].	[MS-FSXTAPI]

2.3 Environment

The following sections identify the context in which the system exists. This includes the systems that use the interfaces provided by this system of protocols, other systems that depend on this system, and, as appropriate, how components of the system communicate.

2.3.1 Dependencies on This System

Microsoft® SharePoint® 2010 Products and Technologies can, but are not required to, use the Microsoft® FAST™ Search system to provide search features.

2.3.2 Dependencies on Other Systems/Components

The Microsoft® FAST™ Search system depends on the following components:

- The Windows Server® 2008 operating system with Service Pack 2 (SP2)
- Microsoft® SQL Server® 2008
- The Microsoft® .NET Framework version 3.5
- Microsoft® Visual C++ 2008 SP1 Redistributable Package (x64)
- Windows® Identity Foundation (WIF)
- Windows® Internet Explorer® 8

The query processing service, as described in section 2.1.1.7, uses WIF to perform authorization operations. To authenticate users and search results, the system uses the security information contained in the end user's **claims** which was generated when the user authenticated to the system.

The browser engine service uses Internet Explorer 8 to render and extract content from Web pages. The system typically extracts items from several types of content sources, such as file servers, Web servers, and database systems. It is possible to extract items from other types of content sources, as described in [MS-FSCF].

2.4 Assumptions and Preconditions

None.

2.5 Use Cases

The following use cases are provided to facilitate understanding of the Microsoft® FAST™ Search system overall. They are not intended to provide a thorough and complete model of the system for any implementation. Each use case has two actors, the user, who submits search queries to and receives search results from the system, and the search administrator, who defines aspects of the user's search experience, such as which items are best bets.

2.5.1 Add Items

This use case describes how to add items to the system to ensure that they are processed, indexed, and, after an indexing latency period, searchable. This process typically occurs when preparing the system for use. Items can be fed into the system either once, to ensure that the content can be searched, or repeatedly, to update existing items with new content.

The actor is a search administrator. The administrator is responsible for ensuring that content can be searched and is therefore also responsible for initiating the transfer of items from content sources into the system. The administrator can transfer items manually or as a regularly scheduled task.

The search administrator starts the indexing connector, which in turn controls the rest of the process, as shown in the following diagram. The Web crawler service is an example of an indexing connector.

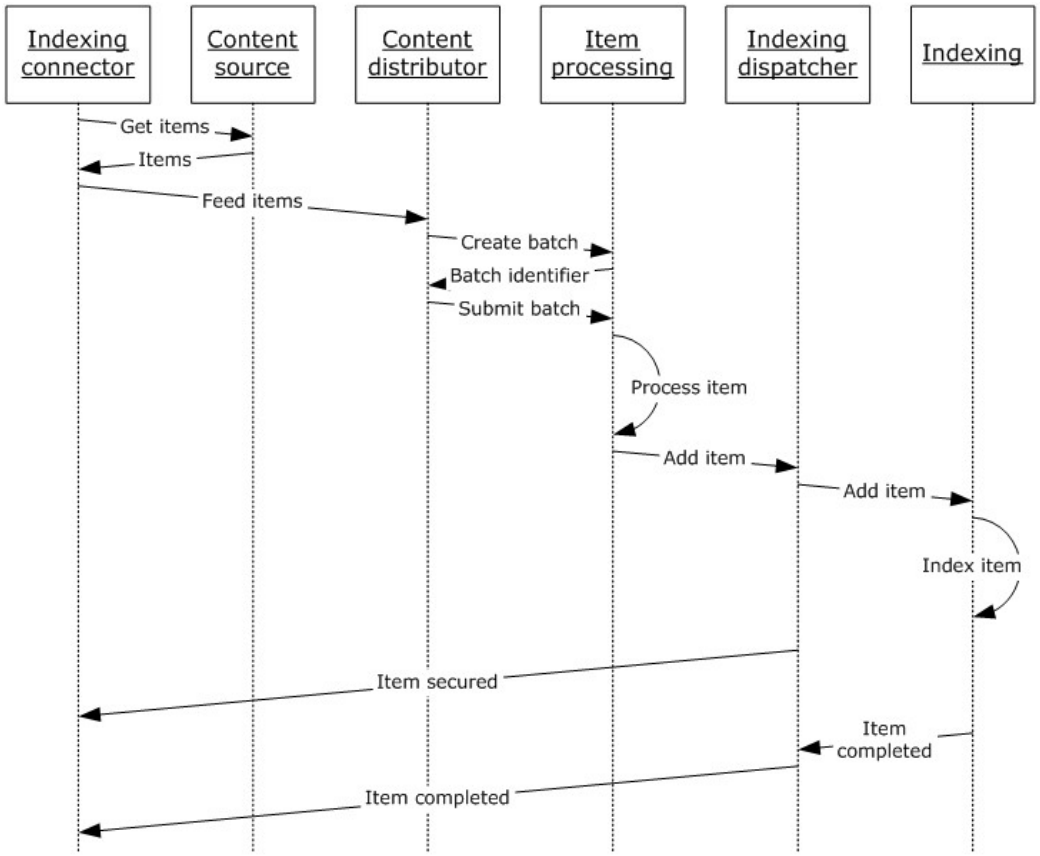

Figure 13: Sequence diagram for adding items to the system

Precondition

The system is fully operational.

Steps

1. An indexing connector retrieves items from a content source and submits them to the content distributor.

2. The content distributor batches the items and sends those batches to the item processing service.

3. The item processing service processes each item in the batch and sends the results to the indexing dispatcher.

4. The indexing dispatcher sends the batch to an indexing node, which then indexes the batch and stores it on a disk.

5. After the items are stored on a disk, the system issues a callback to indicate that the items were stored.

6. The system continues to process the items and creates a new inverted index.

7. The system copies the new inverted index to a query matching node and issues callbacks to indicate that the items are now searchable.

Errors

If an error occurs, the system notifies the indexing connector by issuing a callback.

Post-conditions

- The items are stored on a disk.
- A new inverted index was created.
- The system returns one or more of the items as a query hit if a query matches one or more items.

2.5.2 Perform a Query

This use case describes how a user performs a search query and views the results of that query. This process can occur as part of a larger workflow in which a user is performing a task. To find a specific item or information, the user can perform this process repeatedly and by using different query terms.

The actor is a user who submits queries to the system by using a protocol client, such as a front-end Web server. Typically, the user is an employee of an organization that deployed the system on an intranet. The user is authenticated by the system and primarily searches, browses, or refines search results for content that is hosted on that intranet.

Alternatively, the user might be external to an organization that deployed the system, such as a potential customer of the organization. In this scenario, the user is either authenticated or remains anonymous. In addition, the user in this scenario more typically performs simpler searches. However, this use case assumes that the user's intention is the same as an employee of the organization.

The following diagram illustrates the sequence of events that occur when the user performs a query.

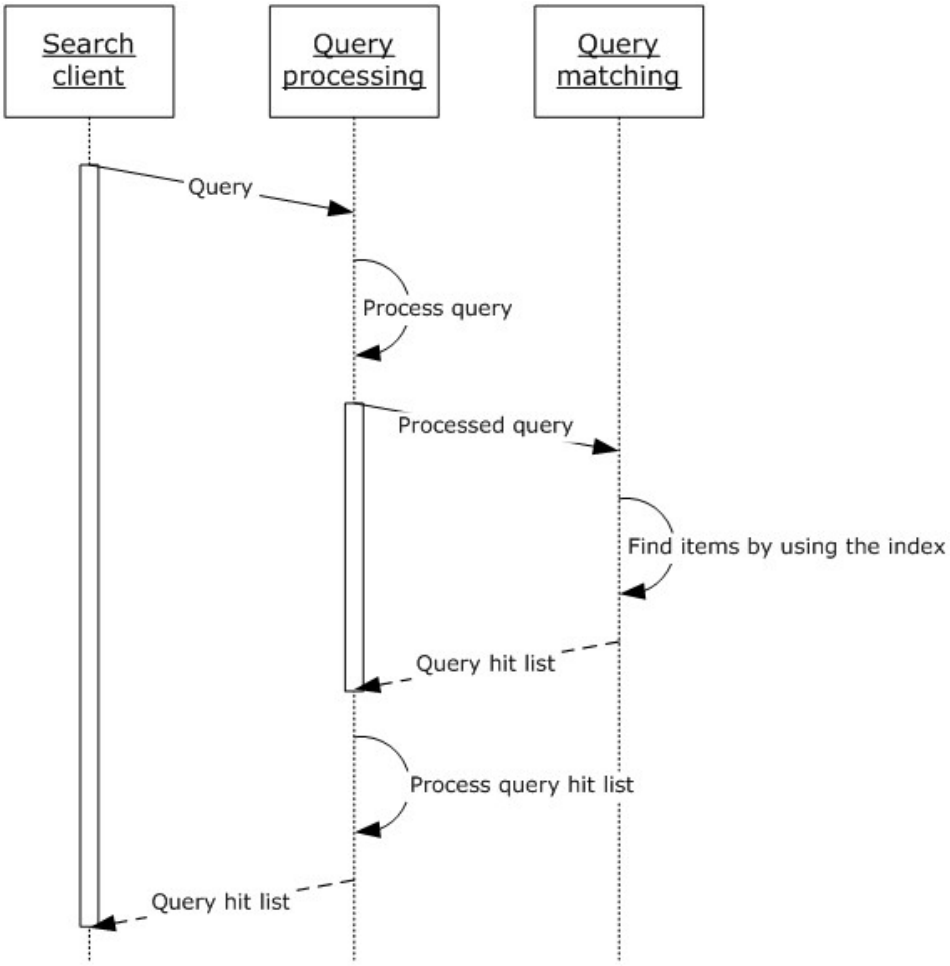

Figure 14: Sequence diagram for performing a query

Preconditions

- The system is fully operational.
- Items were successfully fed into and indexed in the system, as described in section 2.5.1.

Steps

1. The user submits a query to the system by using a protocol client, such as a front-end Web server.
2. The query processing service processes the query and passes it to the query matching service.
3. If the query matches any of the items that are in the inverted index and can be accessed by the user, the query matching service returns those items to the query processing service. The items are organized as a list of query hits in a ranked order.
4. The query processing service processes the query hits and returns them to the protocol client.
5. The protocol client displays the list of query hits to the user.

Errors

If an error occurs, the system notifies the protocol client with an error message.

Post-conditions

- The system is fully operational.
- The system logged the query in the query log.

2.5.3 Add a Synonym

This use case describes how a search administrator uses the administration component to add a synonym to the system. The use of synonyms ensures that the specified synonym is treated as the equivalent of another word in both the items that are added to the system and the queries that are submitted by users. This is one way to refine the configuration of the system to improve search results. For example, if a search administrator wants to improve search results for the keyword "sales", the administrator might determine that many users search for the keyword "figures" and add "figures" as a synonym for "sales".

The actor is a search administrator who is responsible for the overall search experience.

The following diagram illustrates the sequence of events that occur when an administrator adds a synonym.

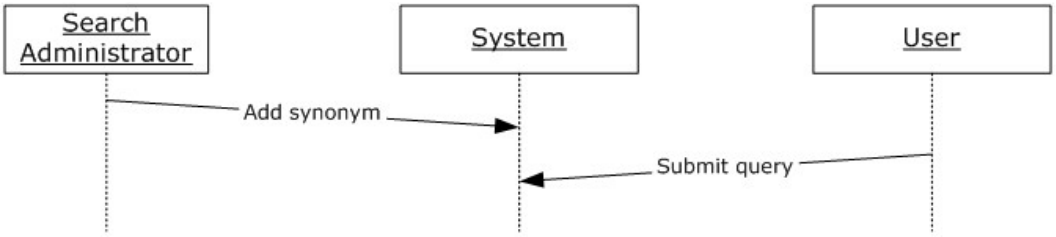

Figure 15: Sequence diagram for adding a synonym

Precondition

The system is fully operational.

Steps

1. A search administrator adds a synonym to the system by using a protocol client, as described in [MS-FSAS].
2. By using a protocol client, a user submits a query to the system, as described in section 2.5.2.
3. The query processing service rewrites the query by adding the synonym, and then sends the rewritten query to the query matching service.
4. The query matching service returns a list of query hits. The list includes items that contain the text of the query, as entered by the user, and items that contain synonyms associated with text in the query, as entered by the user.

Errors

If an error occurs, the system notifies the protocol client with an error message.

Post-condition

The system is fully operational.

2.6 Versioning, Capability Negotiation, and Extensibility

None.

For information about versioning, capability negotiations, and the extensibility of a specific protocol within the system, see the specification for that protocol.

2.7 Error Handling

If the Microsoft® FAST™ Search system is deployed across multiple computers or virtual servers, some services might be temporarily unavailable while individual computers or virtual servers are taken out of operation, restarted, or maintained at the operating-system level. These events generate protocol-specific errors, as described in the specifications for protocols within the system.

Two scenarios can cause system-level errors. An error can occur in the feeding chain and prevent submitted items from being indexed. In addition, an error can occur during query processing and prevent a result from being returned to an indexing connector.

An indexing connector receives callbacks from the item processing service, as described in [MS-FSCF]. If a callback indicates that an error occurred, the indexing connector needs to resubmit the item or the batch of items that failed.

Applications send queries to the query processing service and receive results, as described in [MS-FSQR]. If an error occurs during query processing, that error is reflected in the results that are returned by the query processing service.

2.8 Coherency Requirements

This system has no special coherency requirements.

2.9 Security

Security implications for the Microsoft® FAST™ Search system derive primarily from the following practices:

- Users typically submit queries by using a front-end Web server.
- Search administrators configure and manage content feeding mechanisms, and perform various administrative tasks.

To help ensure the security of the system, these practices necessitate constraints such as the following:

- Only authenticated search administrators can feed content into the system.
- Only authenticated search administrators can change search configurations.
- Unauthenticated users cannot retrieve any content from the system.
- Authenticated users can view information for only those items that they are authorized to access and retrieve from content sources.

- System and search administrators need to be trusted. The system is not designed to prevent malicious actions by administrators.
- Protocol servers need to be deployed and managed in environments that are configured to help prevent attacks and unauthorized access to those servers.
- Administration applications on protocol servers need to be managed in environments that are configured to help prevent unauthorized access to those applications and servers.

To help protect against some security issues, the system implements the following measures:

- The query processing service enforces authorization constraints when users perform queries.
- The system uses **Internet Protocol security (IPsec)** to authenticate communications between different computers and virtual servers. The person who installs the system needs to configure IPsec before starting the installation process for the FAST Search system.
- To gain access to a service API, a user or a calling process entity needs to belong to a local group named "SPSearchExtendedAdministrators".

If the system is configured to use IPsec, all of the computers within the system become part of the same IPsec server farm. Therefore, the addition of a new computer requires IPsec to be reconfigured to add that new computer to the server farm. In addition, the firewall on the new computer needs to be configured the same way as existing computers in the server farm.

2.10 Additional Considerations

None.

3 Examples

The examples in the following sections provide more information about the use and operation of the Microsoft® FAST™ Search system, especially interactions between system components. This document provides the following examples:

- **Add Items** (section 3.1)
- **Perform a Query** (section 3.2)
- **Add a Synonym** (section 3.3)

Each example includes information such as system-level processing that occurs, roles and components that interact with each other, and the state of the system.

For examples of how a specific protocol can be used, see the specification for that protocol.

3.1 Add Items

This example describes the process for adding items to the system by demonstrating how items are fed into, processed, and indexed by the system. It corresponds to the "Add Items" use case that is described in section 2.5.1.

First, an indexing connector sends the following HTML-based items to a content distributor component that is part of the index processing service.

http://example/doc1.html

```
<html xmlns="http://www.w3.org/TR/REC-html40">
<head>
<title>Red car</title>
</head>
<body>
We saw a red car.
</body>
</html>
```

http://example/doc2.html

```
<html xmlns="http://www.w3.org/TR/REC-html40">
<head>
<title>Blue car</title>
</head>
<body>
We also saw a blue car.
</body>
</html>
```

http://example/doc3.html

```
<html xmlns="http://www.w3.org/TR/REC-html40">
<head>
<title>Red automobile</title>
</head>
<body>
```

```
The red automobile was new.
</body>
</html>
```

The item processing service then processes the items by performing the following tasks:

- Detect the language.
- Normalize characters, as described in [MS-FSIN].
- Extract words.
- Find word delimiters for performing tokenization and **stemming**.
- Create output that uses the FAST Index Markup Language (FIXML), as described in [MS-FSFIXML].

The following example is part of the FIXML output for the first item, http://example/doc1.html.

```
<document lang="en-us">
<catalog name="bscpxml" xml:lang="space"></catalog>
<catalog name="bcatcontent">
    <context name="bconf1"><![CDATA[We saw a red car.]]</]}>
    </context>
</catalog>

<catalog name="bt1">
</catalog>

<catalog name="bi1">
</catalog>

<catalog name="meta">
</catalog>

<summary class="content">
    <sField name="contented"><![CDATA[http://example/doc1]]</sField>
    <sField name="collection"><![CDATA[sp]]</sField>
    <sField name="bsrctitle"><![CDATA[Red car]]</sField>
    <sField name="bsrcbody"><![CDATA[We saw a red car.]]</sField>
</summary
</document>
```

The indexing service then creates the requisite index structures according to the index schema. One of the index structures contains, for every term, the internal **item identifier** of each item in which a specific term occurs, similar to the following:

```
a 1,2
    again 3
    also 2
    automobile 3
    blue 2
    car 1,2
    new 3
    red 1,2,3
    saw 1,2
```

```
the 3
was 3
```

The indexing service also creates an item summary for each item. The following example is the item summary for the first item, http://example/doc1.html.

```
Contentid:        http://example/doc1.html
title: Red car
static teaser:    We saw a red car.
```

3.2 Perform a Query

This example describes the process for performing a query that is submitted by a user. It corresponds to the "Perform a Query" use case that is described in section 2.5.2.

By using a protocol client on a front-end Web server, the user enters a query that contains the following terms:

```
car red
```

The protocol client rewrites the query to use the FAST Query Language (FQL), as described in [MS-FSFQL]. The result is the following:

```
and(car, red)
```

Next, the query processing service forwards the FQL version of the query to the query matching service. The query matching service then returns a list of query hits that contains matching items. It does so by using the inverted index to find a list of items that matches each term. The item identifiers for each term are the following:

```
car: 1,2
red: 1,3
```

The query matching service then intersects the lists of item identifiers to produce a complete list of matching items, as follows:

```
1
```

Next, each item is ranked and the list of query hits is sorted according to the rank order of the items. To rank the items, the query matching service uses the item identifiers to look up the properties of each item. Each item identifier includes the static rank score for the item, the total length of the item, and the number of occurrences of each of the query terms in the item. Other properties, such as the item title, indexing time, and author name can also be returned.

Finally, the query matching service sends a list that contains these properties to the query processing service, as described in [MS-FSDQE]. The query processing service then processes the list and returns the results, as described in [MS-FSQR].

3.3 Add a Synonym

This example describes the process for adding a synonym to the system. It corresponds to the "Add a Synonym" use case that is described in section 2.5.3. Before the synonym is added, the sample query returns only one match. After the synonym is added, the sample query returns two matches.

By using a protocol client on a front-end Web server, the user enters a query that contains the following terms:

```
car red
```

The protocol client rewrites the query to use the FAST Query Language (FQL), as described in [MS-FSFQL]. The result is as follows:

```
and(car, red)
```

Next, the query processing service forwards the FQL version of the query to the query matching service. The query matching service then returns a list of query hits that contains one matching item, as follows:

```
http://example/doc1: We saw a red car.
```

To improve search results, a search administrator adds the word "automobile" as a synonym for the word "car".

The user resubmits the query that contains the following terms:

```
car red
```

The protocol client rewrites the query to use the FAST Query Language (FQL), as described in [MS-FSFQL]. The result is as follows:

```
and(car, red)
```

Next, the query processing service rewrites the query by adding the synonym, and then sends the rewritten query to the query matching service. A matching item can now contain either "car" or "automobile", as indicated in the following example.

```
and(any(car,automobile), red)
```

The query matching service returns a list of query hits that contains two matching items, as follows:

```
http://example/doc1.html: We saw a red car.
http://example/doc3.html: The red automobile was new.
```

4 Microsoft Implementations

There are no variations in the behavior of the Microsoft® FAST™ Search system in different versions of FAST Search beyond those described in the specifications of the protocols supported by the system, as listed in section 2.2.

The information in this document is applicable to the following versions of FAST Search:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted in the following section.

4.1 Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

5 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

6 Index