# [MS-FSADS]: Administration Database Schema

**Tools.** The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

| Date | Revision History | Revision Class | Comments |
|---|---|---|---|
| 11/06/2009 | 0.1 | Major | Initial Availability |
| 02/19/2010 | 1.0 | Minor | Updated 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 | 1.04 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 11/15/2010 | 1.04 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 12/17/2010 | 1.04 | No change | No changes to the meaning, language, or formatting of the technical content. |

# Table of Contents

# 1 Introduction

This document specifies the Administration Database Schema, which is used for persisting the information associated with operations in the Administration Services Protocol.

## 1.1 Glossary

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

**best bet**
**crawl component**
**crawled property**
**dictionary**
**document summary**
**fallback managed property**
**featured content**
**field importance level**
**full-text index field**
**index partition**
**index schema**
**keyword**
**managed property**
**one-way synonym**
**promoted document**
**promoted expression**
**promoted location**
**promotion**
**property extraction**
**quality rank**
**query context**
**rank boost**
**rank profile**
**refinement bin**
**refiner**
**search setting**
**search setting context**
**search setting context expression**
**search setting group**
**stemming**
**T-SQL (Transact-Structured Query Language)**
**two-way synonym**

The following terms are specific to this document:

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 1.2 References

### 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative 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.

[MSDN-TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference", http://msdn.microsoft.com/en-us/library/ms189826(SQL.90).aspx

[MS-FSAS] Microsoft Corporation, "Administration Services Protocol Specification", November 2009.

[PROPVARIANT] Microsoft Corporation, "PROPVARIANT", http://msdn.microsoft.com/en-us/library/aa380072.aspx

[RFC2046] Freed, N., and Borenstein, N., "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996, http://ietf.org/rfc/rfc2046.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

### 1.2.2 Informative References

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

[MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification", February 2008.

## 1.3 Structure Overview (Synopsis)

This document describes the database schema used as a persistent store for the methods described in the Administration Services Protocol [MS-FSAS]. These methods create instances of the various configuration concepts related to **keyword**, **dictionary**, and **index schema** management, instances that are retrieved, modified, or deleted through other methods associated with the Administration Services Protocol. A single method in the Administration Services Protocol Server typically results in one or more calls to the database, as specified in the following figure.

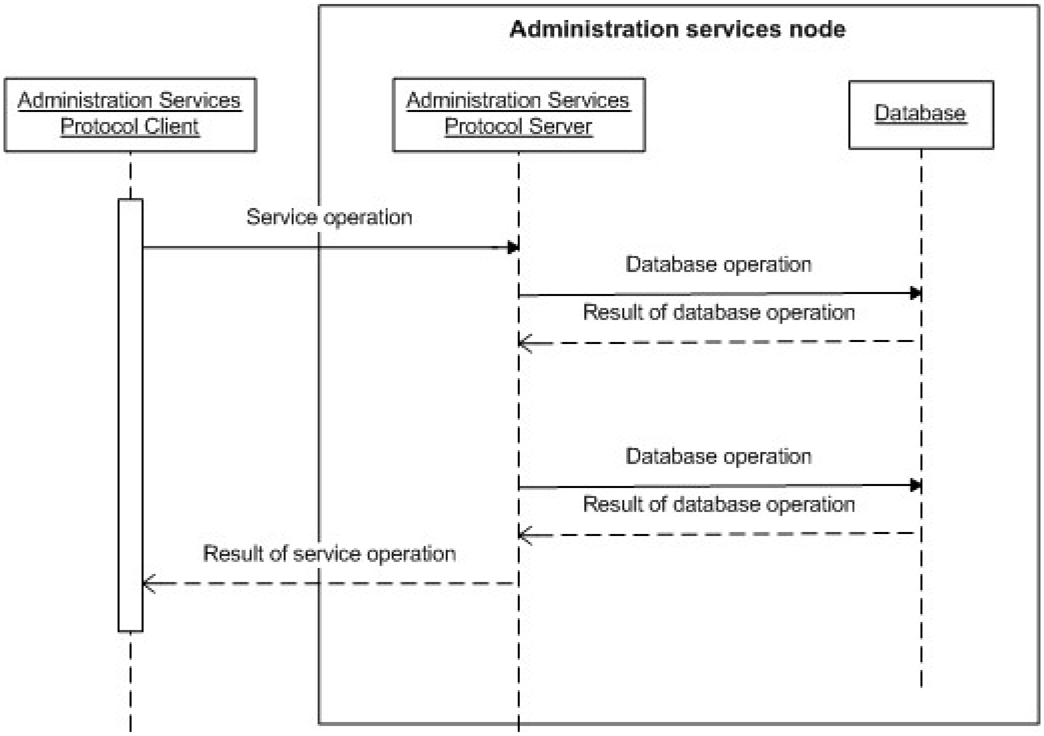
Administration services node
Administration Services Protocol Client
Administration Services Protocol Server
Database
Service operation
Database operation
Result of database operation
Database operation
Result of database operation
Result of service operation

**Figure 1: Flow of messages between the administration services and the database**

In addition to providing persistent storage of these configuration items, the database maintains referential integrity between these items as they are created, modified, or deleted. The communication between the Administration Services Protocol Server and the database is performed by executing **T-SQL (Transact-Structured Query Language)** statements over the protocol described in the Tabular Data Stream Protocol Specification [MS-TDS].

### 1.3.1 Keyword Management Domain Concepts

The purpose of keyword management is to identify words or phrases that are important for the search system. By identifying and configuring these keywords, a site administrator determines business rules that impact the query results when the keyword is used in a query.

The following UML-diagram describes the relationships between the domain concepts of keyword management.

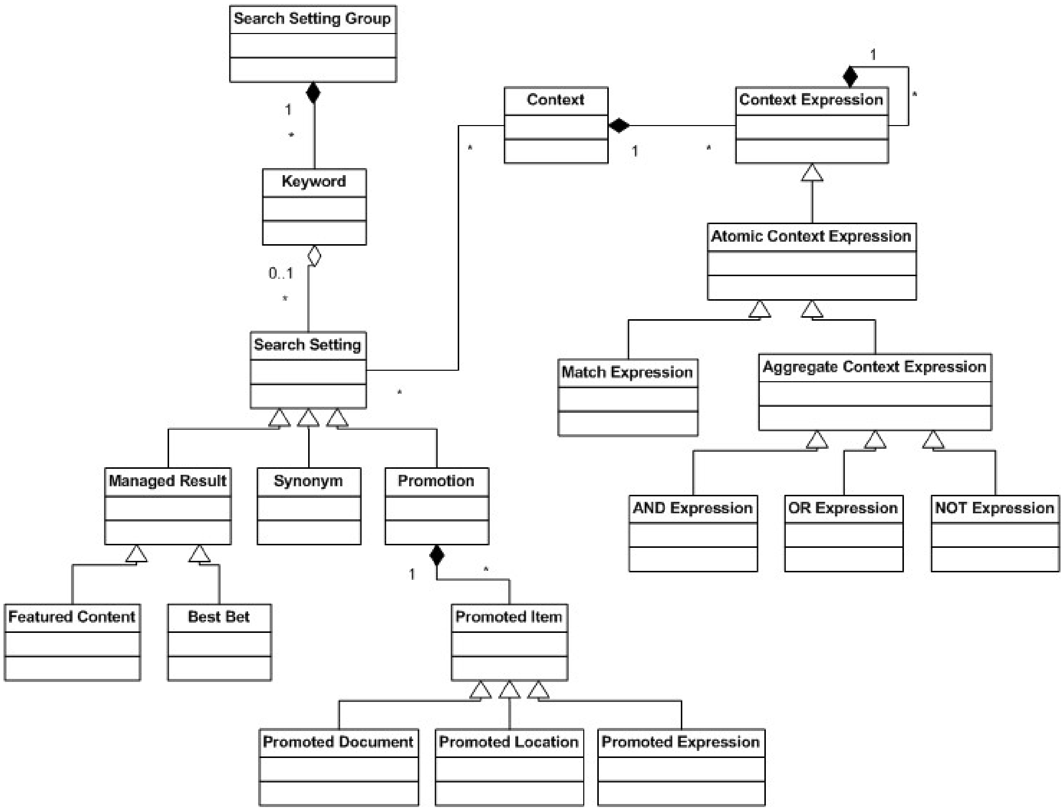

**Figure 2: Keyword management concepts and their relationships**

A **search setting group** contains a collection of keywords. A keyword and its associated **search settings** control how a search query submitted to a search system is rewritten when a query that uses that keyword is executed. Search settings are of type synonyms, **featured content**, **best bets**, or **promotion**. The search settings are applied by associating them with one or more **search setting contexts**. Each search setting context contains a logical expression represented by one or more **search setting context expression** nodes. A search setting is applied to a keyword only if the **query context** matches at least one search setting context expression associated with the search setting of the keyword.

### 1.3.2 Dictionary Management Domain Concepts

A dictionary is a list of entries that consists of a term and an optional description of the term. A term, which must be unique within the dictionary, can either be a single word or a phrase. The persisted dictionary service specified in the Administration Services Protocol [MS-FSAS] section 3.4 provides a means to create, update, retrieve, and delete dictionaries and dictionary entries in the database. The search system uses dictionaries for spell-checking and **property extraction**. Multiple dictionaries may exist, for example to support different languages.

### 1.3.3 Index Schema Management Domain Concepts

The schema service, specified by the Administration Services Protocol [MS-FSAS] section 3.9, contains methods that can be used to manage metadata and relevance information for the search system. The data used by the schema service is stored in the database described by this document.

The schema management service manages metadata by mapping the contents of **crawled properties** to the contents of **managed properties**. The **crawl component** discovers crawled properties during the process of feeding content to the search system. It groups them into categories so that it can manage them at the category level and at the per crawled property level. The schema management service then performs the mapping. Managed properties are part of the search user experience. The schema service control properties on managed properties specify settings for **stemming**, sorting, **document summaries**, and other tasks.

Relevance management is the process of grouping managed properties into **full-text index fields** and managing the set of **rank profiles** used by the search system. A full-text index filed groups the contents of managed properties and enables this content to be searchable. The search system applies **field importance level** and other relevance settings to each managed property. A rank profile is a group of relevance settings that the search system leverages to influence the order of search results when searching one or more full-text index fields.

## 1.4 Relationship to Protocols and Other Structures

None.

## 1.5 Applicability Statement

The database schema described by this document is designed to provide data storage for an implementation of the Administration Services Protocol [MS-FSAS].

## 1.6 Versioning and Localization

None.

## 1.7 Vendor-Extensible Fields

None.

# 2 Structures

The following figures specify the database tables for keyword, dictionary and index schema management, including the individual columns as well as an indication of the foreign key constraints between the tables. The data types and constraints are specified in later sections of this specification.

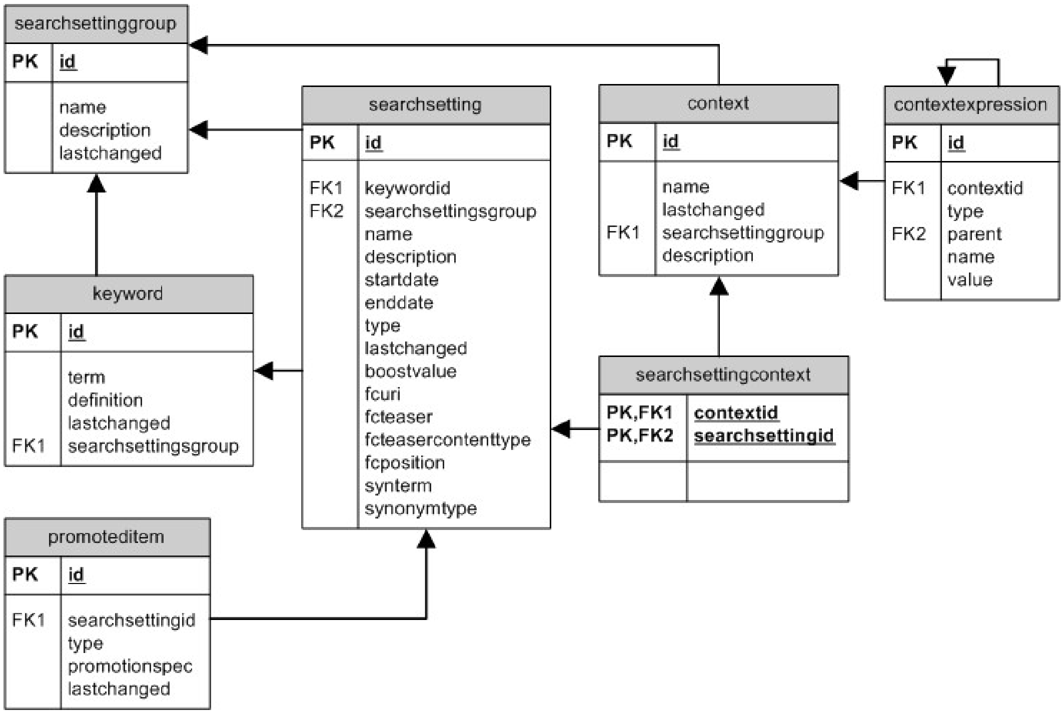

**Figure 3: Database tables for keyword management**

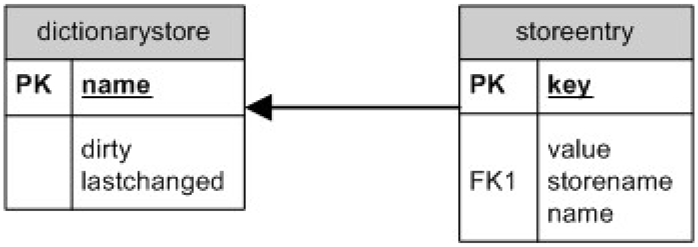

**Figure 4: Database tables for dictionary management**

Release: Sunday, December 19, 2010

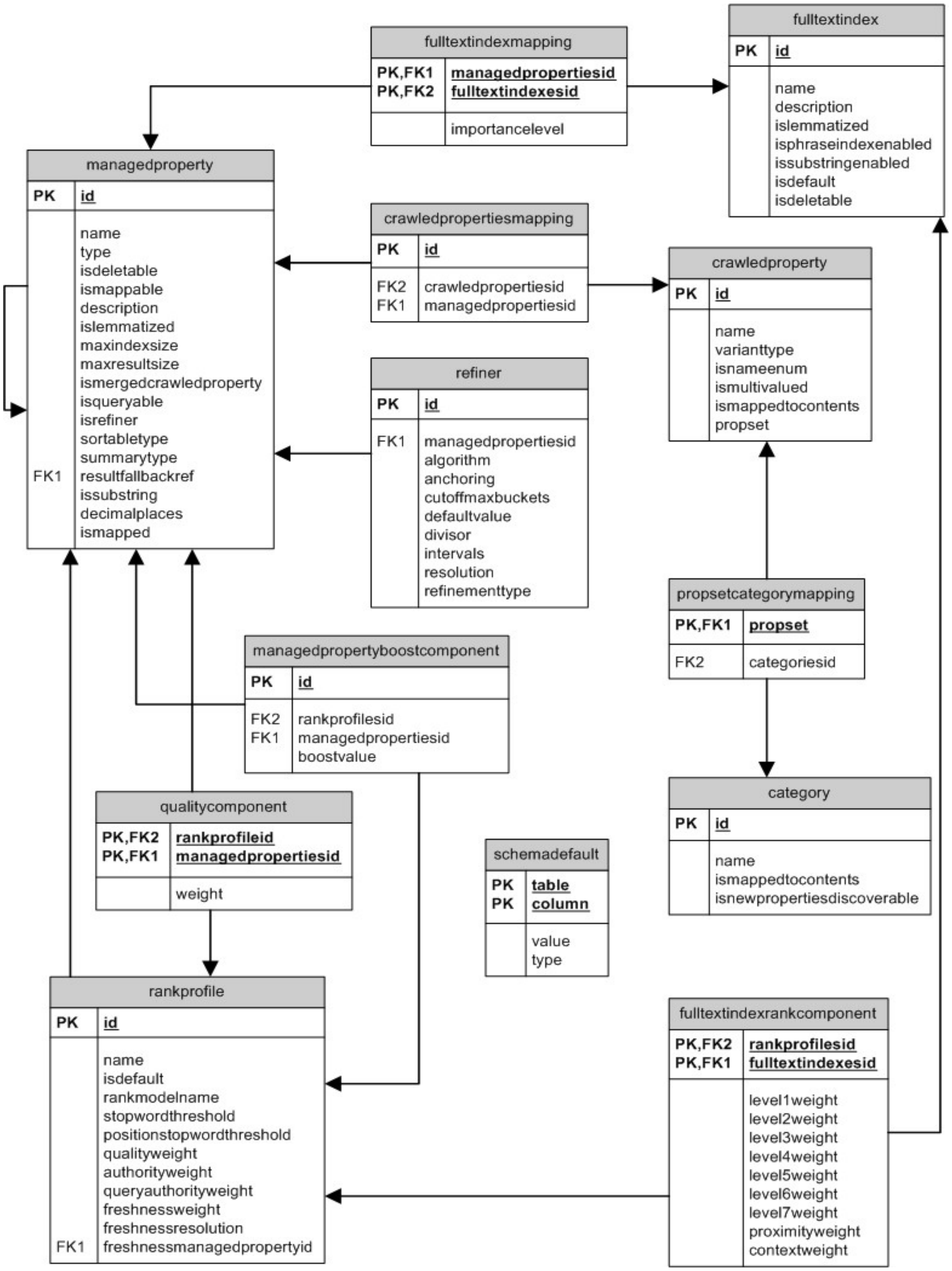

**Figure 5: Database tables for index schema management**

## 2.1 SearchSettingGroup Table

This contains information about the search setting groups. The T-SQL syntax for the table is as follows. T-SQL is specified in the T-SQL Language Reference [MSDN-TSQL-Ref].

```
CREATE TABLE admin.searchsettinggroup (
   id bigint IDENTITY NOT NULL
      CONSTRAINT PK_searchsettinggroup PRIMARY KEY,
   name nvarchar(256),
   description nvarchar(max),
   lastchanged datetime NOT NULL DEFAULT getutcdate());
```

**id:** A unique identifier assigned to the search setting group.

**name:** The unique name of the search setting group.

**description:** The description of the search setting group.

**lastchanged:** The time when the name or description of the search setting group was last changed.

## 2.2 Keyword Table

This stores information about keywords. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.keyword (
id bigint IDENTITY NOT NULL CONSTRAINT PK_keyword PRIMARY KEY,
term nvarchar(256) NOT NULL,
definition nvarchar(max),
lastchanged datetime NOT NULL DEFAULT getutcdate(),
searchsettinggroup bigint NOT NULL
);
```

**id:** A unique identifier assigned to the keyword.

**term:** The keyword term.

**definition:** The description of the keyword.

**lastchanged:** The time when the keyword or any of its search settings were changed.

**searchsettinggroup:** The identifier of the search setting group to which this keyword belongs.

## 2.3 SearchSetting Table

This stores information about search settings. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.searchsetting (
id bigint identity NOT NULL CONSTRAINT PK_searchsetting PRIMARY KEY,
keywordid bigint NOT NULL,
searchsettinggroup bigint NOT NULL,
name nvarchar(256) NOT NULL,
description nvarchar(max),
startdate datetime,
enddate datetime,
type tinyint NOT NULL,
lastchanged datetime NOT NULL DEFAULT getutcdate(),
boostvalue int NOT NULL DEFAULT 0,
fcuri nvarchar(1024),
fcteaser nvarchar(max),
fcteasercontenttype nvarchar(256),
fcposition smallint,
synterm nvarchar(256),
synonymtype tinyint
);
```

**id:** A unique identifier assigned to the search setting.

**keywordid:** The identifier of the keyword to which this search setting belongs.

**searchsettinggroup:** The identifier of the search setting group to which this search setting belongs.

**name:** The name of the search setting.

**description:** The description of the search setting.

**startdate:** The date on which the search setting became valid. If the value is **null**, the search setting becomes valid immediately.

**enddate:** The expiration date of the search setting. If the value is **null**, the search setting never expires.

**type:** The type of the search setting. MUST be a value specified in the following table.

| Value | Meaning |
|---|---|
| 1 | search setting is a synonym |
| 2 | search setting is a promotion |
| 3 | search setting is a best bet |
| 4 | search setting is featured content |

**lastchanged:** The time when the search setting was last changed. This includes promotions for the search setting.

**boostvalue:** The **rank boost** value of a search setting of type promotion. MUST be set when the *type* field is 2.

**fcuri:** The URI of a search setting of type featured content or best bet. MUST be set when the *type* field is 3 or 4.

**fcteasercontenttype:** The content type of the *fcteaser* field associated with a search setting of type featured content or best bet. This field MUST be set if the *type* is 3 or 4 and the *fcteaser* is not **null**. The value of this field is any valid MIME Content type as specified in Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types [RFC2046].

**fcteaser:** Content that can be displayed to a user as part of a best bet or featured content. MAY be set when the *type* is 3 or 4.

**synterm:** The synonym term for a search setting of type synonym. MUST be set when the *type* is 1.

**synonymtype:** The synonym expansion type for a synonym search setting. This field is not used if the *type* field is not set to a value of 1. MUST be one of the values specified in the following table.

| Value | Meaning |
|---|---|
| 0 | synonym is a **one-way synonym** |
| 1 | synonym is a **two-way synonym** |

**fcposition:** The rank position in the result set for a search setting of type featured content or best bet.

## 2.4 PromotedItem Table

This stores information about **promoted documents**, **promoted locations**, and **promoted expressions**. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.promoteditem (
id bigint IDENTITY NOT NULL CONSTRAINT PK_promoteditem PRIMARY KEY,
searchsettingid bigint NOT NULL,        type tinyint NOT NULL,        promotionspec
nvarchar(2048) NOT NULL,    lastchanged datetime NOT NULL DEFAULT getutcdate()
);
```

**id:** A unique identifier assigned to the entry.

**searchsettingid:** The search setting to which this promotion applies.

**type:** The type of the promotion. MUST be one of the values specified in the following table:

| Value | Meaning |
|---|---|
| 8 | entry is a promoted document |
| 9 | entry is a promoted location |
| 10 | entry is a promoted expression |

**promotionspec:** The promotion specification. If the *type* field is set to 8, this MUST be a document identifier. If the *type* field is set to 9 this MUST be a URI. If the *type* field is set to 10 this MUST be a query expression.

**lastchanged:** The time when the entry was last changed.

## 2.5 SearchSettingContext Table

This maintains the many-to-many relationship between search settings and search setting contexts. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.searchsettingcontext (
contextid bigint NOT NULL,
searchsettingid bigint NOT NULL,
CONSTRAINT PK_searchsettingcontext
PRIMARY KEY (contextid, searchsettingid)
);
```

**searchsettingid**: The identifier of a search setting.

**contextid**: The identifier of a context.

## 2.6 Context Table

This contains information about search setting contexts. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.context (
    id bigint IDENTITY NOT NULL CONSTRAINT PK_context PRIMARY KEY,
    name nvarchar(256) NOT NULL,
    lastchanged datetime NOT NULL DEFAULT getutcdate(),
    searchsettinggroup bigint NOT NULL,
    description nvarchar(max)
);
```

**id**: A unique identifier assigned to the *context* entry.

**name**: The name of the *context* entry.

**lastchanged**: The time when the *context* entry or its associated context expression was last changed.

**searchsettinggroup**: The identifier of the search setting group to which the *context* entry belongs.

**description**: The description of the *context* entry.

## 2.7 ContextExpression Table

This contains information about search setting context expression nodes of search setting context expressions. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.contextexpression (
id bigint IDENTITY NOT NULL CONSTRAINT PK_contextexpression PRIMARY KEY,
contextid bigint NOT NULL,
type tinyint NOT NULL,
parent bigint,
name nvarchar(256),
value nvarchar(256)
);
```

**id:** A unique identifier assigned to the context expression node.

**contextid:** The identifier of the *context* entry that is associated with this context expression node.

**type:** The type of the context expression node. MUST be one of the values specified in the following table:

| Value | Meaning |
|---|---|
| 0 | Node is an **AND** node |
| 1 | Node is a **NOT** node |
| 2 | Node is an **OR** node |
| 3 | Node is a **MATCH** node |

**parent:** MUST be set to the identifier of the parent node unless the entry represents a root node. For root nodes the value must be **null**.

**name:** MUST be set to the key of the key-value pair if *type* is 3. Otherwise, MUST be **null**.

**value:** MUST be set to the value of the key-value pair if *type* is 3. Otherwise, MUST be **null**.

## 2.8 DictionaryStore Table

This contains information about dictionaries. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.dictionaryStore (
name nvarchar(256) NOT NULL CONSTRAINT PK_dictionarystore PRIMARY KEY,
dirty bit NOT NULL,
lastchanged datetime NOT NULL DEFAULT getutcdate()
);
```

**name:** The name of the dictionary.

**dirty:** A bit that specifies whether the dictionary was changed. The protocol client sets and resets this bit for implementation-specific purposes.

**lastchanged:** The time when the dictionary was most recently changed.

## 2.9 StoreEntry Table

This table contains dictionary entries. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.storeentry (
"key" nvarchar(256) NOT NULL,
value nvarchar(256),
storename nvarchar(256) NOT NULL,
CONSTRAINT PK_storeentry PRIMARY KEY ("key", storename)
);
```

**key:** The dictionary term.

**value:** An optional description of the value.

**storename:** The name of the dictionary in which this term is located.

## 2.10 RankProfiles Table

This table contains information about rank profiles that the protocol client uses to query one or more full-text index fields. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.rankprofiles (
    id bigint IDENTITY NOT NULL CONSTRAINT PK_rankprofiles PRIMARY KEY,
    name varchar(256) NOT NULL,
    isdefault bit NOT NULL,
    rankmodelname varchar(256) NOT NULL,
    stopwordthreshold bigint NOT NULL,
    positionstopwordthreshold bigint NOT NULL,
    qualityweight int NOT NULL,
    authorityweight int NOT NULL,
    queryauthorityweight int NOT NULL,
    freshnessweight int NOT NULL,
    freshnessresolution int NOT NULL,
    freshnessmanagedpropertyid bigint NOT NULL
);
```

**id:** The identifier of the rank profile.

**name:** The unique name of the rank profile.

**isdefault:** Specifies whether this rank profile is the default.

**rankmodelname:** The name of the rank model that this rank profile extends. The rank model is an implementation specific profile controlling the detailed rank parameters. The default rank model is named "default".

**stopwordthreshold:** Specifies the maximum number of occurrences of a search term in a result set that will contribute to the rank score of documents. If the term appears more often than specified by the threshold, additional occurrences will be ignored. A low value gives better search performance but less result set relevance. A high value decreases search performance, but improves result set relevance.

**positionstopwordthreshold:** Specifies the maximum number of occurrences of a search term in a result set that will contribute to the proximity component of the rank score of documents.

**qualityweight:** Specifies the assigned importance of a document. The quality of a document is query-independent because quality metrics are assigned to individual documents or groups of documents.

**authorityweight:** Specifies the importance of a document as determined by the links from other documents to the document in question.

**queryauthorityweight:** The query authority weight for this rank profile.

**freshnessweight:** Specifies the age of a document compared to the time when the query was issued.

**freshnessresolution:** Specifies the resolution at which freshness values are used for computing rank scores of documents. The following table specifies valid values for this field.

| Value | Meaning |
|---|---|
| 0 | Second resolution |
| 1 | Minute resolution |
| 2 | Hour resolution |

| Value | Meaning |
|---|---|
| 3 | Day resolution |
| 4 | Year resolution |

**freshnessmanagedpropertyname:** The managed property that represents the freshness.

## 2.11 SchemaDefaults Table

This table contains defaults that are used to create managed properties, full-text index fields, refiners, rank profiles, crawled properties, categories, or full-text index field rank components. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.schemadefaults (
    "table" varchar(256) NOT NULL,
    "column" varchar(256) NOT NULL,
    value varchar(1024) NOT NULL,
    type int NOT NULL,
    CONSTRAINT PK_storeentry PRIMARY KEY ("table", "column")
);
```

**table:** A string that specifies the name of a table in this database schema.

**column:** A string that specifies the name of a column in the table specified by the table field.

**value:** A string that specifies a value that will be inserted in the column specified by the table and column fields.

**type:** An integer that specifies the type of the value specified by the value field. Valid values for this field are specified by the following table:

| Value | Meaning |
|---|---|
| 0 | String value |
| 1 | Integer value |
| 2 | Boolean value |

## 2.12 FullTextIndexRankComponents Table

This table contains information about full-text index field rank components. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.fulltextindexrankcomponents (
    rankprofilesid bigint NOT NULL,
    fulltextindexesid bigint NOT NULL,
    level1weight int NOT NULL,
    level2weight int NOT NULL,
    level3weight int NOT NULL,
    level4weight int NOT NULL,
    level5weight int NOT NULL,
    level6weight int NOT NULL,
    level7weight int NOT NULL,
    proximityweight int NOT NULL,
    contextweight int NOT NULL,
    CONSTRAINT PK_fulltextindexrankcomponents PRIMARY KEY (
        rankprofilesid, fulltextindexesid
    )
);
```

**rankprofilesid:** The identifier of the rank profile to which the full-text index field belongs.

**fulltextindexesid:** The identifier of the full-text index field to which this rank component belongs.

**level1weight:** Specifies the rank boost contribution for field importance level 1 members.

**level2weight:** Specifies the rank boost contribution for field importance level 2 members.

**level3weight:** Specifies the rank boost contribution for field importance level 3 members.

**level4weight:** Specifies the rank boost contribution for field importance level 4 members.

**level5weight:** Specifies the rank boost contribution for field importance level 5 members.

**level6weight:** Specifies the rank boost contribution for field importance level 6 members.

**level7weight:** Specifies the rank boost contribution for field importance level 7 members.

**contextweight:** Context weight is related to assigning relevancy weights to different document fields inside a full-text index field.

**proximityweight:** Proximity weight relates to the distance between query terms in the matching documents.

## 2.13 QualityComponents Table

This table contains information about **quality rank** components. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.qualitycomponents (
    rankprofileid bigint NOT NULL,
    managedpropertiesid bigint NOT NULL,
    weight int NOT NULL,
    CONSTRAINT PK_qualitycomponents PRIMARY KEY (
        rankprofileid, managedpropertiesid
    )
);
```

**rankprofileid:** The identifier of the rank profile to which this quality component is mapped.

**managedpropertiesid:** The identifier of the managed property to which this quality component is mapped.

**weight:** The weight of the quality component.

## 2.14 FullTextIndexes Table

This table contains information about full-text index fields. A full-text index field is a grouping of managed properties that a protocol client uses to query these managed properties. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.fulltextindexes (
    id bigint IDENTITY
        CONSTRAINT PK_fulltextindexes PRIMARY KEY NOT NULL,
    name varchar(256) NOT NULL,
    description varchar(1024),
    islemmatized bit,
    isphraseindexenabled bit,
    issubstringenabled bit,
    isdefault bit,
    isdeletable bit
);
```

**id:** The identifier of the full-text index field.

**name:** The unique name of the full-text index field.

**description:** A optional description of the full-text index field.

**islemmatized**: Specifies whether the full-text index field is stemmed.

**isphraseindexenabled:** Not used.

**issubstringenabled:** Specifies whether the full-text index field supports substring queries.

**isdefault:** Specifies whether this full-text index field is the default.

**isdeletable:** Specifies whether this full-text index field can be deleted.

## 2.15 FullTextIndexMappings Table

This table contains mappings between managed properties and full-text index fields. The only properties that can be queried are the managed properties that are mapped to a full-text index field. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.fulltextindexmappings (
    importancelevel int NOT NULL,
    managedpropertiesid bigint NOT NULL,
    fulltextindexesid bigint NOT NULL,
    CONSTRAINT PK_fulltextindexmappings PRIMARY KEY (
        managedpropertiesid, fulltextindexesid
    )
);
```

**importancelevel:** Importance of the referenced managed property within a full-text index field.

**managedpropertiesid:** The identifier of the managed property.

**fulltextindexesid:** The identifier of the full-text index field.

## 2.16 ManagedProperties Table

This table contains information about managed properties. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.managedproperties (
    id bigint IDENTITY
        CONSTRAINT PK_managedproperties PRIMARY KEY NOT NULL,
    name varchar(256) NOT NULL,
    type int NOT NULL,
    isdeletable bit,
    ismappable bit,
    description varchar(1024),
    islemmatized bit,
    maxindexsize bigint,
    maxresultsize bigint,
    ismergedcrawledproperty bit,
    isqueryable bit,
    isrefiner bit,
    sortabletype int,
    summarytype int,
    resultfallbackref bigint,
    issubstring bit,
    decimalplaces int,
    ismapped bit
);
```

**id:** The identifier of the managed property.

**name:** The unique name of the managed property.

**type:** The type of managed property as specified in the Administration Services Protocol Specification [MS-FSAS] section 2.2.4.20.1.

**isdeletable:** Specifies whether the managed property can be deleted.

**ismappable:** Specifies whether it is possible to map new crawled properties to the managed property.

**description:** An optional description of the managed property.

**islemmatized:** Specifies whether the managed property is stemmed.

**maxindexsize:** A value that specifies the maximum number of kilobytes that are indexed for a managed property.

**maxresultsize:** Specifies the maximum size of the query result.

**ismergedcrawledproperty:** Specifies whether this managed property has more than one crawled property mapped to it.

**isqueryable:** Specifies whether this managed property can be queried.

**isrefiner:** Specifies whether this managed property is associated with a **refiner** object.

**sortabletype:** Specifies how this managed property is sorted.

**summarytype:** Specifies whether this managed property produces a summary, as specified in the Administration Services Protocol Specification [MS-FSAS] section 2.2.4.20.2.

**resultfallbackref**: A reference to the **fallback managed property**.

**issubstring:** Specifies whether this managed property is substring query optimized.

**decimalplaces:** Specifies the number of decimal places to persist for this managed property.

**ismapped:** Specifies whether any crawled properties have been mapped to the managed property.

## 2.17 ManagedPropertyBoostComponents Table

This table contains information about managed property rank boost components. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.managedpropertyboostcomponents (
    id bigint IDENTITY
        CONSTRAINT PK_managedpropertyboostcomponents PRIMARY KEY NOT NULL,
    rankprofilesid bigint NOT NULL,
    managedpropertiesid bigint NOT NULL,
    boostvalue varchar(2048) NOT NULL,
);
```

**id:** The identifier of the managed property rank boost component entry.

**rankprofilesid:** The identifier of the rank profile to which this rank boost component is mapped.

**managedpropertiesid:** The identifier of the managed property to which this rank boost component is mapped.

**boostvalue:** The rank boost value.

## 2.18 CrawledPropertiesMappings Table

This table contains mappings between crawled properties and managed properties. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.crawledpropertiesmappings (
    id bigint IDENTITY
        CONSTRAINT PK_crawledpropertiesmappings PRIMARY KEY NOT NULL,
    crawledpropertiesid bigint NOT NULL,
    managedpropertiesid bigint NOT NULL,
);
```

**id:** The identifier of the crawled properties mapping entry.

**crawledpropertiesid:** The identifier of the crawled property.

**managedpropertiesid:** The identifier of the managed property.

## 2.19 CrawledProperties Table

This table contains information about crawled properties. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.crawledproperties (
    id bigint IDENTITY
        CONSTRAINT PK_crawledproperties PRIMARY KEY NOT NULL,
    name varchar(440) NOT NULL,
    varianttype int NOT NULL,
    isnameenum bit,
    ismultivalued bit,
    ismappedtocontents bit,
    propset uniqueidentifier NOT NULL
);
```

**id:** The identifier of the crawled property.

**name:** The name of the crawled property.

**varianttype:** Specifies the data type of the associated crawled property. Valid values for this field are the following values specified for the VARTYPE field of the PROPVARIANT structure [PROPVARIANT]:

| Value | Name |
|---|---|
| 0 | VT_EMPTY |
| 1 | VT_NULL |
| 2 | VT_I2 |
| 3 | VT_I4 |
| 4 | VT_R4 |
| 5 | VT_R8 |
| 6 | VT_CY |
| 7 | VT_DATE |
| 8 | VT_BSTR |
| 9 | VT_DISPATCH |
| 10 | VT_ERROR |
| 11 | VT_BOOL |
| 12 | VT_VARIANT |
| 13 | VT_UNKNOWN |
| 14 | VT_DECIMAL |
| 16 | VT_I1 |
| 17 | VT_UI1 |

| Value | Name |
|---|---|
| 18 | VT_UI2 |
| 19 | VT_UI4 |
| 20 | VT_I8 |
| 21 | VT_UI8 |
| 22 | VT_INT |
| 23 | VT_UINT |
| 30 | VT_LPSTR |
| 31 | VT_LPWSTR |
| 34 | VT_FILETIME |

**isnameenum:** Specifies whether the crawled property name is associated with an enumeration.

**ismultivalued:** Specifies whether the crawled property contains multiple values.

**ismappedtocontents:** Specifies whether the crawled property is mapped to a managed property.

**propset:** Specifies the category to which to map this crawled property.

## 2.20 PropsetCategoryMappings Table

This table contains mappings between crawled properties and categories. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.propsetcategorymappings (
    propset uniqueidentifier
        CONSTRAINT PK_propsetcategorymappings PRIMARY KEY NOT NULL,
    categoriesid bigint NOT NULL
);
```

**propset:** A GUID that identifies the propset.

**categoriesid:** The identifier of the category to which the propset belongs.

## 2.21 Categories Table

This table contains information about categories. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.categories (
    id bigint IDENTITY
        CONSTRAINT PK_categories PRIMARY KEY NOT NULL,
    name varchar(256) NOT NULL,
    ismappedtocontents bit,
    isnewpropertiesdiscoverable bit,
);
```

**id:** The identifier of the category.

**name:** The unique name of the category.

**ismappedtocontents:** Specifies whether the category is mapped to content properties.

**isnewpropertiesdiscoverable:** Specifies whether new crawled properties are reported when documents are processed.

## 2.22 Refiners Table

This table contains information about refiner objects. The T-SQL syntax for the table is as follows:

```
CREATE TABLE admin.refiners (
    id bigint IDENTITY
        CONSTRAINT PK_refiners PRIMARY KEY NOT NULL,
    managedpropertiesid bigint NOT NULL,
    algorithm int,
    anchoring int,
    cutoffmaxbuckets bigint,
    defaultvalue varchar(512),
    divisor float,
    intervals int,
    resolution int,
    refinementtype int
);
```

**id:** The identifier of the refiner.

**managedpropertiesid:** The identifier of the managed property to which this refiner is mapped.

**algorithm:** Contains a reference to the refiner algorithm, as specified in the Administration Services Protocol Specification [MS-FSAS] section 2.2.4.21.1.

**anchoring:** Contains a reference to the refiner anchoring option, as specified in the Administration Services Protocol Specification [MS-FSAS] section 2.2.4.21.2.

**cutoffmaxbuckets:** Specifies a limit for the number of unique values to return for a refiner within an **index partition**.

**defaultvalue:** Specifies a numeric or string value that internally represents an empty value for the field.

**divisor:** Specifies the divisor used to scale values before being displayed to the user. This value is only applicable for numeric refiners.

**intervals**: Specifies the maximum number of **refinement bins** returned by the refiner. This value is only applicable for numeric refiners.

**resolution:** Specifies the resolution of the refiner's refinement bins. For example, use the value of 1000 to align the refinement bins to whole thousand boundaries. This value is only applicable for numeric refiners.

**refinementtype:** The refinement type as specified in the Administration Services Protocol Specification [MS-FSAS] section 2.2.4.21.3.

# 3 Structure Examples

## 3.1 Creating a Search Setting Group

An implementation of the operation **AddSearchSettingGroup** specified in the Administration Services Protocol Specification [MS-FSAS], section 3.8.4.1 could result in the following SQL-statement, where p0 is the name of a search setting group and p1 is the description of the search setting group. The *id* and *lastchanged* fields can be auto-generated by the database.

```
INSERT INTO searchsettinggroup(name, description)VALUES (@p0, @p1)
```

Example values for p0 and p1 are "Advanced Search Settings" and "The settings used for advanced users".

## 3.2 Modifying a Keyword Description

An implementation of the operation **UpdateKeyword** specified in the Administration Services Protocol Specification [MS-FSAS], section 3.8.4.12 could result in the following SQL-statement, where p0 is the identifier of the keyword, p1 the identifier of the search setting group and p2 the new description. The *lastchanged* field can be auto-generated by the database.

```
UPDATE keyword SET definition = @p2WHERE (id = @p0) AND (searchsettinggroup = @p1)
```

Example values for p0, p1 and p2 are 1, 2 and "A statement of what a thing is.", where 1 is the identifier of an existing keyword and 2 is the identifier of an existing search setting group.

### 3.2.1 Creating a Synonym

An implementation of the operation **AddSynonym** specified in the Administration Services Protocol Specification [MS-FSAS], section 3.8.4.31 could result in the following SQL-statement, where p0 is the identifier of the keyword, p1 the identifier of the search setting group, p2 the name of the synonym search setting, p3 is the description, p4 the first date from which the synonym is valid, p5 the last date the synonym is valid, p6 is the synonym term, and p7 the synonym type. The *id* and *last*changed fields can be auto-generated by the database.

```
INSERT INTO searchsetting
(keywordid, searchsettinggroup, name, description, startdate, enddate, type, synterm,
synonymtype)VALUES (@p0, @p1, @p2, @p3, @p4, @p5, 1, @p6, @p7)
```

Example values for the parameters are as follows:

| **Parameter** | **Example value** | **Description** |
|---|---|---|
| p0 | 1 | Identifier of an existing keyword |
| p1 | 2 | Identifier of an existing search setting group |
| p2 | "automobile" | The name of the synonym search setting |
| p3 | "automobile is a synonym of car" | Informative description of the synonym |
| p4 | 2010-01-01 00:00:00 | Activation date |

| Parameter | Example value | Description |
|---|---|---|
| p5 | 2010-02-28 23:59:59 | Deactivation date |
| p6 | "automobile" | The actual synonym |
| p7 | 1 | This is a two-way synonym |

## 3.3 Creating a Crawled Property

An implementation of the operation CreateCrawledProperty specified in the Administration Services Protocol Specification [MS-FSAS] 4.2.4.8 could result in the following SQL statement, where p0 is the name of the crawled property, p1 the GUID of the propset and p2 is the variant type of the crawled property.

```
INSERT INTO crawledproperty(name, propset, varianttype)VALUES (@p0, @p1, @p2)
```

Example values for p0, p1 and p2 are "doc title", 25892e17-80f6-415f-9c65-7395632f0223 and 1.

# 4 Security Considerations

Unauthorized access to the database is prevented by only granting access to the user that runs the Administration Services Protocol Server specified in the Administration Services Protocol Specification [MS-FSAS].

# 5 Appendix A: 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.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

# 6 Change Tracking

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

# 7 Index