[MS-KQL]:
Keyword Query Language Structure Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

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.

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final

documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
01/20/2012	0.1	New	Released new document.
04/11/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the structure of the Keyword Query Language (KQL). KQL is a language for expressing search criteria.

Sections 1.7 and 2 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. All other sections and examples in this specification are informative.

1.1 Glossary

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

Augmented Backus-Naur Form (ABNF)
Coordinated Universal Time (UTC)
Unicode
UTF-8

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

Boolean
dynamic rank
item
managed property
metadata schema
query text
rank
result set
time zone
token

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

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

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.

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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

1.2.2 Informative References

[MS-FQL2] Microsoft Corporation, "Fast Query Language Version 2 Protocol Specification".

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

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

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

1.3 Overview

Application implementers and end users use KQL to express criteria for searching. A typical scenario for using KQL is an application that enables users to search for **item**s and browse through results.

KQL specifies a syntax for search queries that enables users and application implementers to formulate search queries in a structure that resembles natural language and at the same time allows the specification of Boolean matching rules on text and properties of the searched items.

A KQL expression consists of search **token**s, operators, and property restrictions. A search token consists of a value or a range of values to search for, and an operator specifies how to include, exclude, and **rank** the search results. Examples of operators include **AND**, **OR**, **NOT**, **NEAR**, and **XRANK**. A property restriction specifies a Boolean predicate on one property of the searched items.

1.4 Relationship to Protocols and Other Structures

The Search Protocol uses KQL as described in [MS-SEARCH].

An FQL string token supports a KQL mode, FQL is described in [MS-FQL2].

1.5 Applicability Statement

KQL is intended for both application implementers and end users. Application implementers use KQL for searches when they use the Search protocol as described in [MS-SEARCH]. End users typically use KQL for entering search criteria in a search input field in an application.

1.6 Versioning and Localization

None.

1.7 Vendor-Extensible Fields

None.

2 Structures

A KQL expression consists of search tokens, operators, and property restrictions. A search token consists of a value or a range of values to search for, and an operator specifies how to include, exclude, and rank the search results. A property restriction specifies a **Boolean** predicate on one property of the searched items.

KQL operators are case sensitive, and operators use uppercase. Some operators are placed between operands, and other operators are placed before operands. Where noted in the following subsections, operators can have parameters that are placed after the operator in parentheses.

The following words are operators:

- **ALL**
- **AND**
- **ANY**
- **NEAR**
- **NONE**
- **NOT**
- **ONEAR**
- **OR**
- **WORDS**
- **XRANK**

A special class of operators, property operators, is used for property restrictions. The following are property operators:

- **:**
- **=**
- **<>**
- **>**
- **>=**
- **<**
- **<=**

The structure of a KQL expression corresponds to the following rules, which themselves conform to **Augmented Backus-Naur Form (ABNF)** as specified in [RFC5234].

```
kql-expression = (operator-expression / expression-list)

expression-list = (operator-expression operator-expression)
    / (expression-list operator-expression)
```

```
operator-expression = (all / and / any / near / none / not / onear
    / or / words / xrank / basic-expression / paren-expression)

paren-expression = "(" kql-expression ")"

basic-expression = ([qualification] unquoted-string-value)
    / ([qualification] quoted-string-value)
    / property-restriction

; Operator expressions
all = "ALL" "(" 1*string-value ")"
and = operator-expression "AND" operator-expression
any = "ANY" "(" 1*string-value ")"
none = "NONE" "(" 1*string-value ")"
not = "NOT" operator-expression
or = operator-expression "OR" operator-expression

near = operator-expression "NEAR" [proximity-param] operator-expression
onear = operator-expression "ONEAR" [proximity-param] operator-expression
proximity-param = "(" [["N" "="] integer-value] ")"

words = "WORDS" "(" words-param-list ")"
words-param-list = words-param *([","] words-param)
words-param = [qualification] string-value

xrank = operator-expression "XRANK" "(" xrank-param-list ")" operator-expression
xrank-param-list = xrank-param *([","] xrank-param)
xrank-param = ("pb" "=" float-value)
    / ("rb" "=" float-value)
    / ("cb" "=" float-value)
    / ("avgb" "=" float-value)
    / ("stdb" "=" float-value)
    / ("nb" "=" float-value)
    / ("n" "=" integer-value)

; Property restriction
property-restriction = [qualification]
    property-name property-operator property-value
property-name = property-token / quoted-string-value
property-token = 1*(%x30-39 / %x41-5a / %x5f / %x61-7a / %xaa / %xb5 / %xba
    / %xc0-d6 / %xe0-ffffffff)
property-value = property-typed-value
    / unquoted-property-token
    / quoted-string-value
property-operator = ":" / "=" / "<>" / ">" / ">="/ "<" / "<="
unquoted-property-token = 1*(%x01-08 / %x0b-0c / %x0e-1f / %x21 / %x23-27
    / %x2a-3b / %x3f-ffffffff)
property-typed-value = boolean-value / %x22 boolean-value %x22
    / float-value / %x22 float-value %x22
    / integer-value [".." integer-value]
    / %x22 integer-value [".." integer-value] %x22
    / date-named
    / date-value-no-ws [".." date-value-no-ws]
    / %x22 date-value [".." date-value] %x22
date-named = "today" / %x22 "today" %x22
    / "yesterday" / %x22 yesterday %x22
    / %x22 "this week" %x22
    / %x22 "this month" %x22
    / %x22 "last month" %x22
```

```
    / %x22 "this year" %x22
    / %x22 "last year" %x22

; Tokens
boolean-value = "true" / "false"

; The following are culture dependent and are not specified here:
; float-value, integer-value, date-value, date-value-no-ws

string-value = quoted-string-value / unquoted-string-value

; <quoted-string-value> can contain any characters
; except for double quotation marks
quoted-string-value = DQUOTE 1*(%x00-21 / %x23-ffffffff) DQUOTE

; <unquoted-string-value> cannot contain white space,
; double quotation mark, and parentheses.
; <unquoted-string-value> can contain property-chars in the beginning or at
; the end, but not in the middle
unquoted-string-value = *property-chars
    *(%x01-08 / %x0b-0c / %x0e-1f / %x21 / %x23-27 / %x2a-39 / %x3b
    / %x3f-ffffffff)
    *property-chars
property-chars = ":" / "=" / ">" / "<"

; General syntax elements
qualification = "+" / "-"
```

For readability, the preceding rules assume that no extra white space exists in the KQL expression. However, with the exception of **property-operator** (no white space before and after), **qualification** (no white space after), ".." in ranges (no white space before and after), and parameter assignment (no white space before and after **=**), KQL does permit white space to immediately precede and follow parentheses, commas, operators, tokens, and property restrictions.

Also, although ABNF as specified in [RFC5234] does not explicitly support any encoding other than US-ASCII, the **quoted-string-value**, **unquoted-string-value**, **property-token**, and **unquoted-property-token** elements support wide character values that have **UTF-8** encoding.

2.1 Operators

2.1.1 ALL Operator

The **ALL** operator MUST specify one or more token operands separated by white space. To be returned as a match, an item MUST contain all the operands.

2.1.2 AND Operator

The **AND** operator MUST specify two KQL expression operands. To be returned as a match, an item MUST match both operands.

2.1.3 ANY Operator

The **ANY** operator MUST specify one or more token operands separated by white space. To be returned as a match, an item MUST contain at least one of the operands.

2.1.4 NEAR Operator

The **NEAR** operator MUST specify two operands, which in turn MUST each specify an expression to be matched.

If it is specified, the *N* named parameter specifies the maximum number of interspersed, unmatched, indexed tokens. If *N* is not specified, the maximum number is set to 8.

To match the operands of the **NEAR** operator, the item MUST match both expressions, with no more than the specified number of interspersed, unmatched, indexed tokens.

The following MUST be accepted as legal operands of the **NEAR** operator:

- string token (quoted or unquoted)
- **ANY** (section 2.1.3) operator expression
- **OR** (section 2.1.8) operator expression
- **NEAR** operator expression
- **WORDS** (section 2.1.9) operator expression

Other expressions MUST NOT be accepted as legal operands.

If the two operands match the same indexed token, the matches MUST be considered near each other.

2.1.5 NONE Operator

The **NONE** operator MUST specify one or more token operands separated by white space. To be returned as a match, an item MUST NOT contain any of the operands.

2.1.6 NOT Operator

The **NOT** operator MUST specify exactly one KQL expression operand. To be returned as a match, an item MUST NOT match the operand.

2.1.7 ONEAR Operator

The **ONEAR** (ordered near) operator functions in the same way that the **NEAR** operator does (as specified in section 2.1.4) except that the operands MUST match the searched items in the specified order.

For example, an **ONEAR** expression with the string tokens "string1" and "string2" as operands and with the parameter *N* (token distance) set to 1 MUST match "string1 string2", but MUST NOT match "string2 string1".

2.1.8 OR Operator

The **OR** operator MUST specify two KQL expression operands. To be returned as a match, an item MUST match any or both operands.

2.1.9 WORDS Operator

The protocol server MUST support the definition of synonyms in a query string that uses the **WORDS** operator. The **WORDS** operator MUST specify one or more token operands separated by white space or comma. To be returned as a match, an item MUST contain one or more of the operands.

The protocol server MUST ignore the trailing asterisk character in an operand that is a string token prefix.

The protocol server MUST ignore the preceding plus or minus character in an operand that is a qualified token.

2.1.10 XRANK Operator

The **XRANK** operator allows dynamic control over ranking. It boosts the **dynamic rank** of items based on certain term occurrences without changing which items that match the query.

An **XRANK** expression MUST contain one expression operand that MUST be matched (called match expression), and one expression operand (called rank expression) that contributes only to dynamic rank and MUST NOT affect which items are returned as matches. The matching rank expression will add a boost value to the item's total rank.

The named parameters in the following table are valid with the **XRANK** operator:

Named parameter	Default value	Description
cb	0	Specifies the constant boost, corresponds to *a* in the XRANK formula (see section 2.1.10.1).
rb	0	Specifies the range boost, corresponds to *b* in the XRANK formula. This factor is multiplied with the range of rank values in the **result set**.
pb	0	Specifies the percentage boost, corresponds to *c* in the XRANK formula. This factor is multiplied with the item's own rank compared to the minimum value in the result set.
avgb	0	Specifies the average boost, corresponds to *d* in the XRANK formula. This factor is multiplied with the average rank value of the result set.
stdb	0	Standard deviation boost, corresponds to *e* in the XRANK formula. This factor is multiplied with the standard deviation of the rank values of the result set.
nb	0	Normalized boost, corresponds to *f* in the XRANK formula. This factor is multiplied with the product of the variance and average score of the rank values of the result set.
n	0	Number of results to compute statistics from. This parameter does not affect the number of results to which the XRANK contributes; it is just a means to exclude "irrelevant" documents from the statistics calculations.

At least one of the parameters *cb*, *rb*, *pb*, *avgb*, *stdb*, or *nb* MUST be specified.

2.1.10.1 XRANK Formula

The following formula is used for calculating rank values:

$$r_i = a + b \cdot (max - min) + c \cdot (r_i - min) + d \cdot \bar{x} + e \cdot \sigma + f \cdot \frac{\bar{x} \cdot \sigma^2}{\overline{x^2}}$$

where r_i is the rank value of the i^{th} hit,

max (min) is the max (min) rank value of all hits,

$\bar{x}$ is the average rank value of the hits,

σ is the $sqrt(variance)$ of the rank values,

$\overline{x^2}$ is the average of the square of the rank values of the hits

a, b, c, d, e and f are the XRANK parameters

2.1.11 Implicit Operator

The KQL syntax supports a sequence of expressions (the **expression-list** element) without any operator between the expressions. In this case, there is an implicit operator between the expressions. The implicit operator is either **AND** (section 2.1.2) or **OR** (section 2.1.8). Setting the implicit operator is outside the KQL syntax; it is set through the [MS-SEARCH] protocol.

If the query contains any non-property operator (**ALL** (section 2.1.1), **AND** (section 2.1.2), **NOT** (section 2.1.6), **XRANK** (section 2.1.10), and so forth), the query MUST be evaluated as if the implicit operator is **AND** (section 2.1.2).

There are other special cases regarding the use of the implicit operator. See section 2.3.1.1 for the use of the implicit operator in combination with qualified string tokens, and section 2.2.4 for the use of the implicit operator in combination with property restrictions.

2.1.12 Parentheses

Parentheses are used to group subexpressions to change the evaluation order or to make the expression more readable. Parentheses can be nested and are evaluated from inner to outer.

2.1.13 Operator Precedence and Associativity

Operators follow a precedence that defines the evaluation order of expressions containing these operators.

Operators associate with either the expression on their left or the expression on their right; this is called associativity.

The following table shows the precedence and associativity of operators from highest to lowest precedence.

Operator	Associativity
NOT	Right to left
ONEAR	Left to right
NEAR	Left to right
XRANK	Left to right
AND	Left to right

Operator	Associativity
OR	Left to right
Implicit	Left to right

2.2 Property Restrictions

A property restriction specifies a Boolean predicate on one property of the searched items. The protocol server MUST recognize a sequence of characters as a property restriction if it starts with a property name, followed by one of the property operators, followed by a value, without additional characters between name, operator, and value.

If the property name is found as a **managed property** in the **metadata schema**, the type of the value MUST match the type of the managed property. If the property name is not found in the metadata schema, the type of the value SHOULD be handled as a string. The property restriction MUST match the item if the value provided in the query matches the value of the item's property according to the operator.

The operator MUST be one of the following:

Operator	Description
:	The property of the item contains the specified value.
=	The property of the item is equal to the specified value.
<>	The property of the item is not equal to the specified value.
>	The property of the item is greater than the specified value.
>=	The property of the item is greater than or equal to the specified value.
<	The property of the item is less than the specified value.
<=	The property of the item is less than or equal to the specified value.

2.2.1 Property Values

The protocol server MUST support the following types of values: string, Boolean, float, integer, and date. The protocol server MUST accept both quoted and unquoted forms of values.

2.2.2 Property Ranges

For integer and date values the protocol server MUST support ranges. The protocol server MUST interpret range A..B as the set of values from A to B where both A and B are inclusive. For date ranges this means from the beginning of day A to the end of day B.

2.2.3 Property Qualification

If a property restriction (section 2.2) is preceded by a minus character, the protocol server MUST evaluate it the same way as if it was preceded by the **NOT** (section 2.1.6) operator.

If a property restriction is preceded by a plus character, the protocol server MUST ignore the plus character.

2.2.4 Implicit Operator for Property Restrictions

In a sequence of expressions without any operators between the expressions (the **expression-list** element in the ABNF grammar), the following MUST be followed for property restrictions (section 2.2) in the sequence.

Generally, the property restrictions (section 2.2) MUST be interpreted as if **AND** (section 2.1.2) was present between the property restrictions. The following are equivalent:

```
name1:value1 name2:value2
name1:value1 AND name2:value2
```

If the sequence contains two or more property restrictions with the same property name, the property restrictions with the same name MUST be interpreted as if **OR** (section 2.1.8) was present between the property restrictions. The following are equivalent:

```
name1:value1 name1:value2
name1:value1 OR name1:value2
```

An implicit operator used between a property restriction and an expression that is not a property restriction MUST be evaluated as if the **AND** operator was present. The following are equivalent:

```
token1 name1:value1
token1 AND name1:value1
```

2.3 Tokens

2.3.1 String Tokens

A **quoted-string-value** introduces text phrases, which are string values enclosed in double quotes. Any **Unicode** character is allowed with the exception of double quotes. An item MUST match a phrase if it contains all tokens that appear between the quotes, uninterrupted, and in the exact order in which they are specified.

An **unquoted-string-value** introduces unquoted string values. It cannot contain white space characters, double quotes, or parentheses. Also it cannot contain characters that are used for property operators (:, <, >, =) except at the beginning and at the end of the value.

The **unquoted-property-token** used in property restrictions (section 2.2) is similar to an **unquoted-string-value**, but allows some additional characters.

2.3.1.1 Qualified String Tokens

The **quoted-string-value** and **unquoted-string-value** elements can be qualified by a minus or a plus character.

- "+" denotes tokens that MUST be present in an item for a match. These are token inclusions.
- "-" denotes tokens that MUST NOT be present in an item for a match. These are token exclusions.

The exact semantics of inclusions and exclusions depend on whether the implicit operator (see section 2.1.11 for details) is **AND** or **OR**, as specified in sections 2.3.1.1.1 and 2.3.1.1.2.

2.3.1.1.1 Implicit AND operator

The following rules cover the case when the implicit operator is **AND** (section 2.1.2):

- "+" MUST be equivalent to using the **AND** (section 2.1.2) operator.
- "-" MUST be equivalent to using the **AND** and **NOT** (section 2.1.6) operators.

2.3.1.1.2 Implicit OR operator

The following rules cover the case when the implicit operator is **OR** (section 2.1.8):

1. If the query contains any non-property operators (**ALL** (section 2.1.1), **AND** (section 2.1.2), **XRANK** (section 2.1.10), and so on), the query MUST be evaluated as if the implicit operator is **AND (**section 2.3.1.1.1).
2. Otherwise, the evaluation depends on the presence of inclusions.
    1. If there are no inclusions specified, then at least one of the non-qualified tokens MUST match:
        - (exclusions) **AND** (non-qualified tokens)
    2. If there is at least one inclusion specified, then a match on the non-qualified tokens is not required:
        - (exclusions) **AND** ((inclusions) **OR** ((inclusions) **AND** (non-qualified tokens)))

2.3.1.2 String Token Prefix

A string token prefix is a string token that ends with an asterisk character, "*". The "*" MUST be evaluated as a wildcard, that is it matches zero or more characters.

The wildcard evaluation MUST be supported for the elements **quoted-string-value**, **unquoted-string-value**, and **unquoted-property-token**.

2.3.2 Boolean Tokens

Boolean tokens represent logical values and MUST be either "true" or "false".

Boolean tokens MUST be recognized in the following syntactic element:

- Property values where the property name is found as a managed property in the metadata schema of type Boolean or a corresponding type.

In other places Boolean tokens MUST be handled as string tokens.

2.3.3 Integer Tokens

The non-terminal symbol **integer-value** introduces integer values. A protocol server SHOULD take into account the culture in which the **query text** was formulated and recognize the string representation of the integer specific to that culture.

Integer tokens MUST be recognized in the following syntactic elements:

- Property values where the property name is found as a managed property in the metadata schema of type integer or a corresponding type.

- Parameter values to operators where the parameter is of type integer or a type that can be assigned an integer value.

In other places integer tokens MUST be handled as string tokens.

2.3.4 Float Tokens

The non-terminal symbol **float-value** introduces floating point values. A protocol server SHOULD take into account the culture in which the query text was formulated and recognize the string representation of the floating point values specific to that culture.

Float tokens MUST be recognized in the following syntactic elements:

- Property values where the property name is found as a managed property in the metadata schema of type **float** or a corresponding type.
- Parameter values to operators where the parameter is of type **float** or a type that can be assigned a **float** value.

In other places float tokens MUST be handled as string tokens.

2.3.5 Date Tokens

A date token represents a specific date or a date interval. The protocol server SHOULD allow a time part to be present with the date. If a time part is present, the protocol server MUST ignore it.

For all date values, the protocol server SHOULD interpret the date as being specified in a given **time zone**, typically the time zone of the user. Time zone is set through the [MS-SEARCH] protocol. If the time zone is not set or not available, **Coordinated Universal Time (UTC)** SHOULD be assumed.

Date tokens MUST be recognized in the following syntactic element:

- Property values where the property name is found as a managed property in the metadata schema of type date or a corresponding type.

In other places date tokens MUST be handled as string tokens.

The non-terminal symbol **date-value-no-ws** introduces a date token that MUST not contain any white space characters. A protocol server SHOULD take into account the culture in which the query text was formulated and recognize the string representation of dates specific to that culture.

The non-terminal symbol **date-value** introduces a date token that MAY contain white space characters. A protocol server SHOULD take into account the culture in which the query text was formulated and recognize the string representation of dates specific to that culture.

An implementation MUST support names that represent date intervals relative to the current date as follows:

Name of date interval	Description
today	Represents the time from the beginning of the current day until the end of the current day.
yesterday	Represents the time from the beginning of the day until the end of the day that precedes the current day.

Name of date interval	Description
this week	Represents the time from the beginning of the current week until the end of the current week.
this month	Represents the time from the beginning of the current month until the end of the current month.
last month	Represents the entire month that precedes the current month.
this year	Represents the time from the beginning of the current year until the end of the current year.
last year	Represents the entire year that precedes the current year.

The names of date intervals that contain a space MUST be quoted.

3 Structure Examples

3.1 Operator Examples

3.1.1 ALL Operator Example

The following expression matches items that contain all of the terms "cat", "dog", and "fox".

```
ALL(cat dog fox)
```

3.1.2 AND Operator Example

The following expression matches items that contain both "cat" and "dog".

```
cat AND dog
```

3.1.3 ANY Operator Example

The following expression matches items that contain at least one of the terms "cat", "dog", and "fox".

```
ANY(cat dog fox)
```

3.1.4 NEAR Operator Examples

The following expression matches items that contain "cat" and "dog" as long as no more than eight (the default number) indexed tokens separate them.

```
cat NEAR dog
```

The following expressions match items that contain "cat" and "dog" as long as no more than five indexed tokens separate them.

```
cat NEAR(N=5) dog
cat NEAR(5) dog
```

If the operands of the **NEAR** operator match the same indexed token, they are considered near each other. For example, the following expression matches items that contain only the indexed token "cat" because both operands match and are considered near each other, even though both operands match the same indexed token.

```
cat NEAR (cat OR dog)
```

3.1.5 NONE Operator Example

The following expression matches items that contain none of the terms "cat", "dog", and "fox".

```
NONE(cat dog fox)
```

3.1.6 NOT Operator Example

The following expression matches items that do not contain "aardvark".

```
NOT aardvark
```

3.1.7 ONEAR Operator Examples

The following expression matches items that contain "cat" that appear before "dog", as long as no more than eight (the default number) indexed tokens separate them.

```
cat ONEAR dog
```

The following expressions match items that contain "cat" that appear before "dog" as long as no more than five indexed tokens separate them.

```
cat ONEAR(N=5) dog
cat ONEAR(5) dog
```

3.1.8 OR Operator Example

The following expression matches all the items that contain either "cat" or "dog" or both.

```
cat OR dog
```

3.1.9 WORDS Operator Examples

The following expression matches all the items that contain either "TV" or "television" or both.

```
WORDS(TV television)
```

When using the **WORDS** operator, the terms "TV" and "television" are treated as synonyms instead of separate terms. Therefore, instances of either term are ranked as if they were the same term.

Any trailing asterisk character in operands is ignored, so the following are equivalent.

```
WORDS(word1* word2)
WORDS(word1 word2)
```

Any qualification (preceding plus and minus character) for operands is ignored, so the following are equivalent.

```
WORDS(+word1 -"word2 word3")
WORDS(word1 "word2 word3")
```

3.1.10 XRANK Operator Examples

The following expression matches items that contain either "cat" or "dog" or both. The expression boosts the dynamic rank of those items that also contain "thoroughbred". The constant boost is set to 100.

```
(cat OR dog) XRANK(cb=100) thoroughbred
```

The following expression matches items that contain either "cat" or "dog" or both. The expression boosts the dynamic rank of those items that also contain "thoroughbred". The normalized boost is set to 1.5.

```
(cat OR dog) XRANK(nb=1.5) thoroughbred
```

3.1.11 Implicit Operator Examples

The following expression illustrates an implicit operator. There is an implicit **AND** (section 2.1.2) or **OR** (section 2.1.8) operator between "cat" and "dog".

```
cat dog
```

The following expressions are equivalent. The first query contains a non-property operator and the query is evaluated as if the implicit operator is **AND**.

```
cat (dog OR fox)
cat AND (dog OR fox)
```

3.1.12 Parentheses Example

The following expression uses parentheses to change the default evaluation order. It will match items that contain "cat" or "dog", and in addition contain "fox".

```
(cat OR dog) AND fox
```

3.2 Property Restriction Examples

In the following expressions, it is assumed that size is a managed property found in the metadata schema of type integer or a corresponding type. The expressions match items where the size property is equal to, not equal to, less than, or greater than 100, respectively.

```
size=100
size<>100
size<100
size>100
```

3.2.1 Property Range Example

In the following expressions, it is assumed that size is a managed property found in the metadata schema of type integer. The expression matches items where the size property is in the range [100,200].

```
size:100..200
```

3.2.2 Property Qualification Examples

In the following expressions, it is assumed that size is a managed property found in the metadata schema of type integer.

The following are equivalent and match items where the size property is not equal to 100:

```
-size=100
NOT size=100
size<>100
```

The following are equivalent and match items where the size property is equal to 100:

```
size=100
+size=100
```

3.2.3 Implicit Operator for Property Restriction Examples

In the following expressions, it is assumed that author and filetype are managed properties found in the metadata schema of type string.

The following are equivalent:

```
author:"John Smith" filetype:docx
author:"John Smith" AND filetype:docx
```

The following are equivalent:

```
author:"John Smith" author:"Jane Smith"
author:"John Smith" OR author:"Jane Smith"
```

The following are equivalent:

```
cat filetype:docx
cat AND filetype:docx
```

3.3 Token Examples

3.3.1 String Token Examples

Each of the following expressions consists of a single string token.

```
potato
"to be or not to be"
"AND"
true
100
3.14159265358979
2005-12-31
```

The following expression is a property restriction (section 2.2) containing a string token as value. Here it is assumed that filetype is a managed property found in the metadata schema of type string, or that filetype is not found in the metadata schema (the default type of properties is string).

```
filetype:docx
```

3.3.1.1 Qualified String Token Examples

See section 3.3.1.1.1 and section 3.3.1.1.2 for examples where the implicit operator is **AND** (section 2.3.1.1.1) and **OR** (section 2.3.1.1.2), respectively.

3.3.1.1.1 Implicit AND Operator Examples

The following queries match the same items:

```
cat +dog
cat AND dog
```

The following queries match the same items:

```
cat -dog
cat AND NOT dog
```

The following queries match the same items:

```
cat +dog -fox
cat AND dog AND NOT fox
```

3.3.1.1.2 Implicit OR Operator Examples

The following queries match the same items:

```
cat dog +fox
fox OR (fox AND (cat OR dog))
```

The following queries match the same items:

```
cat dog -fox
(NOT fox) AND (cat OR dog)
```

The following queries match the same items:

```
cat +dog -fox
(NOT fox) AND (dog OR (dog AND cat))
```

3.3.1.2 String Token Prefix Example

The following string token matches "cat", "calculator", "calendar", and any other indexed token that begins with "ca" because the "*" character at the end of the string value is evaluated as a wildcard as specified in section 2.3.1.2.

```
ca*
```

3.3.2 Boolean Token Examples

In the following expressions, it is assumed that IsDocument is a managed property found in the metadata schema of type Boolean or a corresponding type.

```
IsDocument:true
IsDocument:false
IsDocument:"true"
IsDocument:"false"
```

3.3.3 Integer Token Examples

In the following expressions, it is assumed that Boost is a managed property found in the metadata schema of type integer or a corresponding type. US English is assumed as the user culture (other cultures can use a different format for integer values).

```
Boost:360
Boost:-25
Boost:"360"
Boost:"-25"
```

The **NEAR** (section 2.1.4) operator accepts an integer value for the parameter *N*.

```
cat NEAR(N=5) dog
```

3.3.4 Float Token Examples

In the following expressions, it is assumed that Factor is a managed property found in the metadata schema of type float or a corresponding type. US English is assumed as the user culture (other cultures can use a different format for float values).

```
Factor:2.71828182846
Factor:-5.3
Factor:"2.71828182846"
Factor:"-5.3"
```

The **XRANK** (section 2.1.10) operator accepts a float value for the parameter *cb*.

```
cat XRANK(cb=1.5) dog
```

3.3.5 Date Token Examples

In the following expressions, it is assumed that Modified is a managed property found in the metadata schema of type date or a corresponding type. US English is assumed as the user culture (other cultures can use a different format for date values).

```
Modified:2008-01-29
Modified:"2008-01-29"
Modified:today
Modified:"this week"
```

4 Security

4.1 Security Considerations for Implementers

None.

4.2 Index of Security Parameters

None.

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® SharePoint® Server 2013 Preview

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