iOS SDK Building Query Predicates - CM

iOS SDK Building Query Predicates - CM

An `NSPredicate` object defines the logical conditions for determining whether a record is a match for a query. The `PWCMEQuery` class supports a subset of the predicate behaviors offered by the full `NSPredicate` class.

ON THIS PAGE

- iOS SDK Building Query Predicates - CM
- Predicate Rules for Query Objects
- Supported Predicate Operators
- Sample Predicate Format Strings
  - Matching a Field to a Specific Value
  - Matching a Field to One or More Values
  - Matching a Field That Starts with a String Value
  - Matching a Field Containing a Tokenized String
  - Matching a Field with Regular Expressions

Predicate Rules for Query Objects

The predicates you create for your query objects must follow these rules:

- Predicates are based on a format string. You cannot use value- or block-based predicates.
- Predicates use only the operators listed in the "Supported Predicate Operators" table below.
- Predicates operate only on fields containing the following types of data:
  - NSString
  - NSDate
  - NSNumber
- Key names used in predicates correspond to fields in the currently evaluated record. Key names may include the names of the record's metadata properties such as "creationDate" or any data fields you added to the record. You cannot use key paths to specify fields in related records.
- Predicates support the following variable substitution strings:
  - Use %@ for value objects such as strings, numbers and dates.
  - Use %K for the name of a field. This substitution variable indicates that the substituted string should be used to look up a field name.
- `BETWEEN` queries are not currently supported.

Supported Predicate Operators

Operation	Supported Operators
Basic comparisons	=, ==, >=, =>, <=, =<, <, >!=, <>
Basic compound predicates	AND, && NOT
String comparisons	BEGINSWITH
Aggregate operations	IN CONTAINS
Regular expressions	MATCHES

Specifying an unsupported operator or data type in your query's predicate results in a silent error when you execute the query.

Sample Predicate Format Strings

Matching a Field to a Specific Value

To match the contents of a field to a specific value, use a predicate similar to the ones shown below. All of the listed predicates generate the same set of results, which in the example means that the "favoriteColors" field contains the value "red". The value in the field must match the value you specify in the predicate exactly. String-based comparisons are case insensitive, but otherwise all comparisons must be an exact match of the specified value.

```
NSPredicate *predicate = nil;
predicate = [NSPredicate
predicateWithFormat:@"ANY favoriteColors =
'red'"];
predicate = [NSPredicate
predicateWithFormat:@"favoriteColors CONTAINS
'red'"];
predicate = [NSPredicate
predicateWithFormat:@"'red' IN
favoriteColors"];
predicate = [NSPredicate
predicateWithFormat:@"%K CONTAINS %@",
@"favoriteColors", @"red"];
```

Matching a Field to One or More Values

You can match against more than one value at a time by using a predicate similar to the ones below. In the example, the predicates report a match if the value in the "favoriteColor" field of a record matches either of the values "red" or "green".

```
NSPredicate *predicate = nil;
predicate = [NSPredicate
predicateWithFormat:@"ANY { 'red', 'green' } =
favoriteColor"];
predicate = [NSPredicate
predicateWithFormat:@"favoriteColor IN { 'red',
'green' }"];
```

Matching a Field That Starts with a String Value

For fields that contain string values, you can match the beginning portion of the string using the `BEGINSWITH` or `CONTAINS` operator as shown below. Currently, you cannot use other string comparison operators such `ENDSWITH`. When using this operator, the field must contain a string value and must start with the string you specified. Matches are *NOT* case insensitive. In the examples, the predicate matches records whose "favoriteColors" field contained the strings "red", "reddish" or "red green duct tape".

```
NSString *matchString = @"red";
NSPredicate predicate = nil;
predicate = [NSPredicate
predicateWithFormat:@"ANY favoriteColors
BEGINSWITH 'red'"]
predicate = [NSPredicate
predicateWithFormat:@"ANY favoriteColors
BEGINSWITH %@", matchString]
predicate = [NSPredicate
predicateWithFormat:@"ANY favoriteColors
CONTAINS 'red'"]
predicate = [NSPredicate
predicateWithFormat:@"ANY favoriteColors
CONTAINS %@", matchString]
```

Matching a Field Containing a Tokenized String

To perform a tokenized search of a record's fields, use the special operator self. A tokenized search searches any fields that have full-text search enabled, which are all string-based fields by default. Below is an example that searches the fields of the record for the token strings "bob" and "smith". Each distinct word is treated as a separate token for the purpose of searching. Comparisons are case- and diacritic-insensitive. These token strings may be found in single or multiple fields, but all of the tokens must be present in a record for it to be considered a match.

```
NSPredicate *predicate = nil;
predicate = [NSPredicate
predicateWithFormat:@"SELF CONTAINS 'bob
smith'"]
```

Matching a Field with Regular Expressions

To perform a search with regular expressions, use the `MATCHES` operator.

```
NSPredicate *predicate = nil;
predicate = [NSPredicate
predicateWithFormat:@"emailID MATCHES
'phunware^'"]
```