MDQuery Reference

Carbon > File Management

2005-06-04

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

MDQuery Reference

Derived From:	CFType
Framework:	CoreServices/CoreServices.h
Declared in	MDQuery.h
Companion guides	Spotlight Overview Spotlight Query Programming Guide

Overview

MDQuery is a CF-compliant object, follows the CF conventions, and can be used with the CF polymorphic functions, such as `CFRetain`. MDQuery encapsulates queries against the System store of the file metadata.

An MDQuery normally executes asynchronously and posts progress notifications as the results are collected. During the gathering phase the query results conform to the specified value lists and sorting.

MDQuery gathers results and processes updates only while the current thread's run loop is running.

For functions that take an MDQueryRef parameter, if this parameter is not a valid MDQueryRef, the behavior is undefined. `NULL` is not a valid MDQueryRef.

For functions that take CF*Ref parameters, such as CFStringRef and CFArrayRef, if this parameter is not a valid CF object of the correct type, the behavior is undefined. `NULL` is not a valid CF*Ref.

Functions by Task

Creating Queries

`MDQueryCreate` (page 9)
Creates a new query instance.

`MDQueryCreateSubset` (page 9)
Creates a new query that is a subset of the specified parentquery.

`MDQuerySetSearchScope` (page 18)
Sets the search scope for a query instance.

Getting and Setting Query Parameters

`MDQueryGetBatchingParameters` (page 12)
Returns the current parameters that control the batching of progress notifications.

`MDQuerySetBatchingParameters` (page 15)
Set the query batching parameters.

`MDQueryCopyValueListAttributes` (page 8)
Returns the list of attribute names for which values are being collected by the query.

`MDQueryCopySortingAttributes` (page 7)
Returns the list of attribute names used to sort the results.

`MDQueryCopyQueryString` (page 7)
Returns the query string of the query.

Setting Callback Functions

`MDQuerySetCreateResultFunction` (page 16)
Sets the function used to create the result objects of the MDQuery.

`MDQuerySetSortComparator` (page 19)
Sets the function used to sort the results of an MDQuery.

`MDQuerySetCreateValueFunction` (page 17)
Sets the function used to create the value objects of the MDQuery.

Starting, Stopping and Pausing Queries

`MDQueryExecute` (page 11)
Run the query, and populate the query with the results.

`MDQueryStop` (page 19)
Stops the query from generating more results.

`MDQueryDisableUpdates` (page 10)
Disables updates to the query result list.

`MDQueryEnableUpdates` (page 10)
Enables updates to the query result list.

`MDQueryIsGatheringComplete` (page 15)
Returns true if the first phase of a query, the initial result gathering, has finished.

Getting Query Result Values

`MDQueryCopyValuesOfAttribute` (page 8)
Returns the list of values from the results of the query for the specified attribute.

`MDQueryGetAttributeValueOfResultAtIndex` (page 12)
Returns the value of the named attribute for the result at the given index.

`MDQueryGetCountOfResultsWithAttributeValue` (page 12)
Returns the number of results which have the given attribute and attribute value.

`MDQueryGetIndexOfResult` (page 13)
Returns the current index of the given result.

`MDQueryGetResultAtIndex` (page 14)
Returns the current result at the given index.

`MDQueryGetResultCount` (page 14)
Returns the number of results currently collected by the query.

Getting the Type Identifier

`MDQueryGetTypeID` (page 15)
Returns the type identifier of all MDQuery instances

Functions

MDQueryCopyQueryString

Returns the query string of the query.

```
CFStringRef MDQueryCopyQueryString (
   MDQueryRef query
);
```

Parameters

query
The query.

Return Value
A CFStringRef containing the query string.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryCopySortingAttributes

Returns the list of attribute names used to sort the results.

```
CFArrayRef MDQueryCopySortingAttributes (
   MDQueryRef query
);
```

Parameters

query
The query.

Return Value
A CFArrayRef containing the attribute names used to sort the query results.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryCopyValueListAttributes

Returns the list of attribute names for which values are being collected by the query.

```
CFArrayRef MDQueryCopyValueListAttributes (
   MDQueryRef query
);
```

Parameters

query

The query.

Return Value
A CFArrayRef containing the attribute names of the collected values.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryCopyValuesOfAttribute

Returns the list of values from the results of the query for the specified attribute.

```
CFArrayRef MDQueryCopyValuesOfAttribute (
   MDQueryRef query,
   CFStringRef name
);
```

Parameters

query

The query.

name

The attribute name to return the value of. If the attribute is not one of those requested when the query was created the behavior is undefined

Return Value
A CFArrayRef containing the value objects for the specified attribute. The array contents are not ordered and contain only one occurrence of each value. The array contents may change over time if the query is configured for live-updates.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryCreate

Creates a new query instance.

```
MDQueryRef MDQueryCreate (
   CFAllocatorRef allocator,
   CFStringRef queryString,
   CFArrayRef valueListAttrs,
   CFArrayRef sortingAttrs
);
```

Parameters

allocator

The CFAllocator object to be used to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

queryString

The query expression string for this query.

valueListAttrs

An optional array of attribute names. The query will collect the values of these attributes into uniqued lists that can be used to summarize the results of the query and allow the user to further qualify the search. This parameter may be `NULL` if no value lists are required. Value list collection increases CPU usage and significantly increases the memory usage of an MDQuery. The attribute names are CFStrings.

sortingAttrs

A n array of attribute names used to sort the results, or `NULL` if no sorting is required. The first name in the array is used as the primary sort key, the second as the secondary key, and so on. The comparison of like-typed values is a simple, literal comparison. Sorting increases memory usage and significantly increases the CPU usage of an MDQuery. It is usually more efficent to allow the MDQuery to sort the results than retrieiving the values and sorting the results yourself. The attribute names are CFStrings.

Return Value

An MDQueryRef, or `NULL` on failure. If the query string is empty or malformed the function returns NULL.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`MDQuery.h`

MDQueryCreateSubset

Creates a new query that is a subset of the specified parentquery.

```
MDQueryRef MDQueryCreateSubset (
   CFAllocatorRef allocator,
   MDQueryRef query,
   CFStringRef queryString,
   CFArrayRef valueListAttrs,
   CFArrayRef sortingAttrs
);
```

Parameters

allocator

The CFAllocator object to be used to allocate memory for the new object. Pass NULL or `kCFAllocatorDefault` to use the current default allocator.

query

The parent query

queryString

The query expression string for this query.

valueListAttrs

An optional array of attribute names. The query will collect the values of these attributes into uniqued lists that can be used to summarize the results of the query and allow the user to further qualify the search. This parameter may be `NULL` if no value lists are required. Value list collection increases CPU usage and significantly increases the memory usage of an MDQuery. The attribute names are CFStrings.

sortingAttrs

A n array of attribute names used to sort the results, or `NULL` if no sorting is required. The first name in the array is used as the primary sort key, the second as the secondary key, and so on. The comparison of like-typed values is a simple, literal comparison. Sorting increases memory usage and significantly increases the CPU usage of an MDQuery. It is usually more efficent to allow the MDQuery to sort the results than retrieiving the values and sorting the results yourself. The attribute names are CFStrings.

Return Value
An MDQueryRef, or `NULL` on failure. If the query string is empty or malformed the function returns NULL.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryDisableUpdates

Disables updates to the query result list.

```
void MDQueryDisableUpdates (
   MDQueryRef query
);
```

Parameters

query

The query.

Discussion
This function should be called before iterating over query results that could change due to live-updates. The disabled state is a counter and disabling can be done recursively and from different threads.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryEnableUpdates

Enables updates to the query result list.

```
void MDQueryEnableUpdates (
   MDQueryRef query
);
```

Parameters

query

The query.

Discussion
This function should be called when finished iterating through the list of results. Live-updates to the query results will continue when all the disables have been matched by a corresponding enable.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryExecute

Run the query, and populate the query with the results.

```
Boolean MDQueryExecute (
   MDQueryRef query,
   CFOptionFlags optionFlags
);
```

Parameters

query

The query to execute.

optionFlags

A bitwise OR of the `MDQueryOptionFlags` to be used by the query.

Return Value
Returns `TRUE` if the query was started, `FALSE` otherwise. Queries cannot be executed more than once.

Discussion
Queries only gather results or process updates while the current thread's run loop is running.

Queries have two phases: the initial gathering phase that collects all currently matching results and a second live-update phase. Updates occur during the live-update phase if a change in a file occurs such that it no longer matches the query or if it begins to match the query. Files which begin to match the query are added to the result list, and files which no longer match the query expression are removed from the result list.

Query notifications are posted within the context of the same thread which executes the query.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryGetAttributeValueOfResultAtIndex

Returns the value of the named attribute for the result at the given index.

```
void * MDQueryGetAttributeValueOfResultAtIndex (
   MDQueryRef query,
   CFStringRef name,
   CFIndex idx
);
```

Parameters

query

The query.

name

The attribute name to return the values of. If the attribute is not one of those requested in the *valueListAttrs* or *sortingAttrs* parameters to one of the query creation functions, the result will be `NULL`.

idx

The index into the query's result list. If the index is negative or is equal to or larger than the current number of results in the query, the behavior is undefined.

Return Value

The value of the attribute, or `NULL` if the attribute doesn't exist for the specidied result.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`MDQuery.h`

MDQueryGetBatchingParameters

Returns the current parameters that control the batching of progress notifications.

```
MDQueryBatchingParams MDQueryGetBatchingParameters (
   MDQueryRef query
);
```

Parameters

query

The query.

Return Value

An MDQueryBatchingParams structure with the current batching parameters.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`MDQuery.h`

MDQueryGetCountOfResultsWithAttributeValue

Returns the number of results which have the given attribute and attribute value.

```
CFIndex MDQueryGetCountOfResultsWithAttributeValue (
   MDQueryRef query,
   CFStringRef name,
   CFTypeRef value
);
```

Parameters

query

The query.

name

The attribute name to return the result count of. If the attribute is not one of those requested in the *valueListAttrs* parameter, the behavior is undefined.

value

The attribute value for which to return the number of results with that value. This parameter may be `NULL`, in which case the number of results that do not contain the specified attribute is returned.

Return Value
The number of results containing that attribute and value.

Discussion
This count may change over time if the query allows live-updates.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryGetIndexOfResult

Returns the current index of the given result.

```
CFIndex MDQueryGetIndexOfResult (
   MDQueryRef query,
   const void *result
);
```

Parameters

query

The query.

result

The result object to search for. If a custom create-result function has been set and this parameter is not a valid result object that the provided callbacks can handle, the behavior is undefined. If a custom create-result function has not been set this parameter must be a valid MDItemRef.

Return Value
The index of the given result, or `kCFNotFound` if the value is not one of the query's existing results. If you provided a custom result creation function result, the result will be objects created by that function.

Discussion
If a result-create function has been set, and the equal callback is non-`NULL`, it will be used to test the query's results against the candidate result.

Note that the index of a result can change over time if the query allows live-updates.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryGetResultAtIndex

Returns the current result at the given index.

```
const void * MDQueryGetResultAtIndex (
   MDQueryRef query,
   CFIndex idx
);
```

Parameters

query

The query.

idx

The index into the query's result list. If the index is negative, or is equal to or larger than the current number of results in the query, the behavior is undefined.

Return Value
Returns the MDItemRef currently at the given index, or if a result-creation function has been set, returns the result returned by that function.

Discussion
This function causes the result object to be created if it hasn't been created already. For performance reasons you should only request objects that you require. If possible, call this function to fetch only the results you need to display or otherwise process.

Note that the index of a particular result can change over time if the query is configured to allow live-updates.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryGetResultCount

Returns the number of results currently collected by the query.

```
CFIndex MDQueryGetResultCount (
   MDQueryRef query
);
```

Parameters

query

The query.

Return Value
The number of results in the query.

Discussion
Note that the number of results in a query will change over time as the query's result list is updated.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryGetTypeID

Returns the type identifier of all MDQuery instances

```
CFTypeID MDQueryGetTypeID (
   void
);
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryIsGatheringComplete

Returns true if the first phase of a query, the initial result gathering, has finished.

```
Boolean MDQueryIsGatheringComplete (
   MDQueryRef query
);
```

Parameters

query

The query.

Return Value
Returns `TRUE` if the first phase of a query has completed, otherwise `FALSE`.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQuerySetBatchingParameters

Set the query batching parameters.

```
void MDQuerySetBatchingParameters (
   MDQueryRef query,
   MDQueryBatchingParams params
);
```

Parameters

query

The query.

params

An MDQueryBatchingParams structure with the batching parameters to set.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQuerySetCreateResultFunction

Sets the function used to create the result objects of the MDQuery.

```
void MDQuerySetCreateResultFunction (
   MDQueryRef query,
   MDQueryCreateResultFunction func,
   void *context,
   const CFArrayCallBacks *cb
);
```

Parameters

query

The query.

func

The callback function the MDQuery will use to create its results, such as those returned by the function `MDQueryGetResultAtIndex`. This parameter may be `NULL`, in which case any previous result creation settings are cancelled and the MDQuery will subsequently produce MDItemRefs. If a function is specified and is not of type `MDQueryCreateResultFunction` or does not behave as a `MDQueryCreateResultFunction` must, the behavior is undefined.

context

A pointer-sized user-defined value, that is passed as the third parameter to the create function. MDQuery does not use this value, does not retain the context in any way, and requires that the context be valid for the lifetime of the query. If the context is not what is expected by the create function, the behavior is undefined.

cb

A pointer to a `CFArrayCallBacks` structure initialized with the callbacks for the query to use to manage the created result objects. A copy of the contents of the callbacks structure is made, so that a pointer to a structure on the stack can be passed in, or can be reused for multiple query creations. Only version 0 of the `CFArrayCallBacks` is supported. The retain field may be `NULL`, in which case the MDQuery will not add a retain to the created results for the query. The release field may be `NULL`, in which case the MDQuery will not remove the query's retain (such as the one it gets from the create function) on the result objects when the query is destroyed. If the *`copyDescription`* field is `NULL`, the query will create a simple description for the result objects. If the *`equal`* field is `NULL`, the query will use pointer equality to test for equality of results. This callbacks parameter itself may be `NULL` in which case it is treated as a valid version 0 structure with all fields NULL. Otherwise, if any of the fields are not valid pointers to functions of the correct type, or this parameter is not a valid pointer to a `CFArrayCallBacks` callbacks structure, the behavior is undefined. If any of the value values returned from the create function is not one understood by one or more of the callback functions, the behavior when those callback functions are used is undefined. For example, if the create function can return `NULL`, then `NULL` must be understood by the callback functions as a possible parameter. The retain and release callbacks must be a matched set, you should not assume that the retain function will be unused or that additional reference counts will not be taken on the created results.

Discussion

If no create function is specified for an MDQuery, the default result objects are MDItemRefs. Results created after the function `MDQuerySetCreateResultFunction` is called are created through the specified create function, but values created before the function was set, or after it is unset, are not modified. It is not advisable to change this function after the function `MDQueryExecute` has been called. The create-result function is called lazily as results are requested from a query, it is not called on all results, and may not be called at all. This avoids the cost of creating potentially hundreds of thousands of what might be temporary objects.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`MDQuery.h`

MDQuerySetCreateValueFunction

Sets the function used to create the value objects of the MDQuery.

```
void MDQuerySetCreateValueFunction (
   MDQueryRef query,
   MDQueryCreateValueFunction func,
   void *context,
   const CFArrayCallBacks *cb
);
```

Parameters

query

The query.

func

The callback function the MDQuery should use to create the value list values, such as those returned by the function `MDQueryCopyValuesOfAttribute`. This parameter may be `NULL`, in which case any previous value creation settings are cancelled and the MDQuery will subsequently produce the default CFTypeRefs. If a function is specified and is not of type `MDQueryCreateValueFunction` or does not behave as a `MDQueryCreateValueFunction` must, the behavior is undefined.

context

A pointer-sized user-defined value, that is passed as the third parameter to the create function. MDQuery does not use this value, does not retain the context in any way, and requires that the context be valid for the lifetime of the query. If the context is not what is expected by the create function, the behavior is undefined.

cb

A pointer to a CFArrayCallBacks structure initialized with the callbacks for the query to use to manage the created value objects. A copy of the contents of the callbacks structure is made, so that a pointer to a structure on the stack can be passed in, or can be reused for multiple query creations. Only version 0 of the `CFArrayCallBacks` is supported. The retain field may be `NULL`, in which case the MDQuery will not add a retain to the created values. The release field may be `NULL`, in which case the MDQuery will do nothing to remove the query's retain (such as the one it gets from the create function) on the value objects when the query is destroyed. If the *copyDescription* field is `NULL`, the query will create a simple description for the value objects. If the *equal* field is `NULL`, the query will use pointer equality to test for equality of values. This callbacks parameter itself may be `NULL` in which case it is treated as a valid version 0 structure with all fields `NULL`. Otherwise, if any of the fields are not valid pointers to functions of the correct type, or this parameter is not a valid pointer to a `CFArrayCallBacks` callbacks structure, the behavior is undefined. If any of the value values returned from the create function is not one understood by one or more of the callback functions, the behavior when those callback functions are used is undefined. For example, if the create function can return `NULL`, then `NULL` must be understood by the callback functions as a possible parameter. The retain and release callbacks must be a matched set, you should not assume that the retain function will be unused or that additional reference counts will not be taken on the created results.

Discussion
Values created after a create function is set will be created using the newly specified function, but existing values are not modified. It is not advisable to change this function after `MDQueryExecute` has been called with the query.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQuerySetSearchScope

Sets the search scope for a query instance.

```
void MDQuerySetSearchScope (
   MDQueryRef query,
   CFArrayRef scopeDirectories,
   OptionBits scopeOptions
);
```

Parameters

query

The query object to modify.

scopeDirectories

A CFArray of CFStringRef or CFURLRef objects which specify where to search. For convenience the `kMDQueryScopeHome`, `kMDQueryScopeComputer` and `kMDQueryScopeNetwork` constants may also be included in the array.

scopeOptions
Additional options for modifying the search. Currently you must pass 0.

Discussion

Availability
Available in Mac OS X v10.4 and later.

Declared In
MDQuery.h

MDQuerySetSortComparator

Sets the function used to sort the results of an MDQuery.

```
void MDQuerySetSortComparator (
   MDQueryRef query,
   MDQuerySortComparatorFunction comparator,
   void *context
);
```

Parameters

query
The query.

comparator
The callback function the MDQuery uses to sort the results list. This parameter may be `NULL` which cancels previous sort comparator settings. If a function is specified and is not of type `MDQuerySortComparatorFunction` or does not behave as a `MDQuerySortComparatorFunction` must, the behavior is undefined.

context
A pointer-sized user-defined value, that is passed as the third parameter to the create function. MDQuery does not use this value, does not retain the context in any way, and requires that the context be valid for the lifetime of the query. If the context is not what is expected by the create function, the behavior is undefined.

Availability
Available in Mac OS X v10.4 and later.

Declared In
MDQuery.h

MDQueryStop

Stops the query from generating more results.

```
void MDQueryStop (
   MDQueryRef query
);
```

Parameters

query
The query.

Discussion
Queries may be executed only once and cannot be restarted. The query will first complete processing any unprocessed results.do. That may trigger a progress notification, so be aware of that if you are stopping a query from within your progress note handler; that is, during this function, a recursive progress and/or finished notification might occur, which might recursively call your notification handler. It is safe to call this function recursively. You would call this function to stop a query that is generating way too many results to be useful, but still want to access the results that have come in so far. If a query is stopped before the gathering phase finishes, it will not report itself as finished, nor will it send out a finished notification.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

Callbacks

MDQueryCreateResultFunction

Callback function used to create the result objects stored and returned by a query.

```
typedef const void * (*MDQueryCreateResultFunction) (
 MDQueryRef query,
 MDItemRef item,
 void *context
);

forthcoming
```

Parameters

query
The query instance.

item
The default MDItemRef for the result.

context
The user-defined context parameter provided to the `MDQuerySetCreateResultFunction` function.

Return Value
The function must return a pointer-sized value that can be managed with the callbacks which were set at the same time the create function was given to the query. The value must be returned with a reference (such as if the retain callback had been called on it), as implied by the Create name. If this function doesn't wish to create a new object it can return the given MDItemRef, but must also return it with a new retain, and the callbacks must be able to handle an MDItemRef as an input value. If this function returns NULL, NULL will be stored for the moment in the query, MDQueryGetResultAtIndex() may return NULL for that result, and the next time the query wants the result, it will call this function again.

Discussion
The function may hold onto the given attribute name and/or value in some other data structure, but must retain them for them to remain valid.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQueryCreateValueFunction

Callback function used to create the value objects stored and returned by a query.

```
typedef const void * (*MDQueryCreateValueFunction) (
 MDQueryRef query,
 CFStringRef attrName,
 CFTypeRef attrValue,
 void *context
);

forthcoming
```

Parameters

query
The query instance.

attrName
The attribute name of the value.

attrValue
The default value of the value.

context
The user-defined context parameter provided in the `MDQuerySetCreateValueFunction` function.

Return Value
The function must return a pointer-sized value that can be managed with the callbacks which were set at the same time the create function was given to the query. The value must be returned with a reference (such as if the retain callback had been called on it), as implied by the Create name. If this function doesn't wish to create a new object, it can return the given CFTypeRef, but must also return it with a new retain, and the callbacks must be able to handle a CFTypeRef as an input value.

Discussion
The function may hold onto the given attribute name and/or value in some other data structure, but must retain them for them to remain valid

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

MDQuerySortComparatorFunction

Callback function used to sort the results of a query.

```
typedef CFComparisonResult (*MDQuerySortComparatorFunction) (
 const CFTypeRef attrs1[],
 const CFTypeRef attrs2[],
 void *context
);

asdfasdfasdfasdfadsf
```

Parameters

query

The query instance.

attrs1

A C array of attribute values for a result. The values occur in the array in the same order and position that the attribute names were passed in the `sortingAttrs` array when the query was created. The values of the attributes will be `NULL` if the attribute doesn't exist for a result or if read access to that attribute is not allowed.

attrs2

A C array of attribute values for a result. The values occur in the array in the same order and position that the attribute names were passed in the `sortingAttrs` array when the query was created. The values of the attributes will be `NULL` if the attribute doesn't exist for a result or if read access to that attribute is not allowed.

context

The user-defined context parameter provided in the function `MDQuerySetSortComparator`.

Return Value

The function must return one of the CFComparisonResults `kCFCompareLessThan`, `kCFCompareEqualTo`, or `kCFCompareGreaterThan`. There is no provision for unordered results. The comparison should be a total order relation and produce the same results for the same inputs.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`MDQuery.h`

Data Types

Batching Parameters

MDQueryBatchingParams

Structure containing the progress notification batching parameters of a MDQuery.

```
typedef struct {
    size_t first_max_num;
    size_t first_max_ms;
    size_t progress_max_num;
    size_t progress_max_ms;
    size_t update_max_num;
    size_t update_max_ms;
} MDQueryBatchingParams;
```

Fields

`first_max_num`

The maximum number of results that can accumulate before the first progress notification is sent. This value is used only during the initial result-gathering phase of a query.

`first_max_ms`

The maximum number of milliseconds that can pass before the first progress notification is sent. This value is advisory, in that the notification will be triggered at some point after `first_max_ms` milliseconds have passed since the query began accumulating results. This value is used only during the initial result-gathering phase of a query.

`progress_max_num`

The maximum number of results that can accumulate before additional progress notifications are sent. This value is used only during the initial result-gathering phase of a query.

`progress_max_ms`

The maximum number of milliseconds that can pass before additional progress notifications are sent. This value is advisory, in that the notification will be triggered at some point after `progress_max_ms` milliseconds have passed since the query began accumulating results. This value is used only during the initial result-gathering phase of a query.

`update_max_num`

The maximum number of results that can accumulate before an update notification is sent. This value is used only during the live-update phase of a query.

`update_max_ms`

The maximum number of milliseconds that can pass before an update notification is sent. This value is advisory, in that the notification will be triggered at some point after `update_max_ms` milliseconds have passed since the query began accumulating results. This value is used only during the live-update phase of a query.

Discussion

The default batching parameters are undefined and subject to change.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`MDQuery.h`

Miscellaneous

MDQueryRef

A reference to a MDQuery object.

```
typedef struct __MDQuery *MDQueryRef;
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`MDQuery.h`

Constants

Query Option Flags

MDQueryOptionsFlags

Specify the execution mode for a query.

```
typedef enum {
    kMDQuerySynchronous = 1,
    kMDQueryWantsUpdates = 4,
} MDQueryOptionFlags;
```

Constants

`kMDQuerySynchronous`

Specifies that a query should block during the initial gather phase. The query's run loop will run in the default mode. If this option is not specified the query function returns immediately after starting the query asynchronously.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

`kMDQueryWantsUpdates`

Specifies that a query should provide live-updates to the result list after the initial gathering phase. Updates occur during the live-update phase if a change in a file occurs such that it no longer matches the query or if it begins to match the query. Files which begin to match the query are added to the result list, and files which no longer match the query expression are removed from the result list. Currently, this option is ignored if the `kMDQuerySynchronous` parameter is specified. This is subject to change, and you should always pass the value appropraite to the required behavior.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

Notifications

kMDQueryDidFinishNotification

Indicates that a query has finished with the initial result-gathering phase.

```
const CFStringRef kMDQueryDidFinishNotification;
```

Constants
`kMDQueryDidFinishNotification`
Posted to indicate that the query has finished the initial result-gathering phase.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

Discussion
The query results list is not updated as a result of this notification.

This notification is only sent to the application's notification center.

kMDQueryDidUpdateNotification

Indicates that a query's results list has change during the live-update phase of a query.

```
const CFStringRef kMDQueryDidUpdateNotification;
```

Constants
`kMDQueryDidUpdateNotification`
Notification posted to indicate that a change has occured to the query's resuls list during the live-update phase of a query's execution.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

Discussion
The info dictionary of the notification can contain `kMDQueryUpdateAddedItems`, `kMDQueryUpdateChangedItems`, and `kMDQueryUpdateRemovedItems` keys.

This notification is only sent to the application's notification center.

kMDQueryProgressNotification

Indicates that a query's results list has change during the initial result-gathering phase of a query.

```
const CFStringRef kMDQueryProgressNotification;
```

Constants
`kMDQueryProgressNotification`
Notification posted to indicate that a change has occurred to the query's results list during the initial result-gathering phase of execution.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

Discussion
New items are typically added during this phase, however it is possible for items to be removed or updated, if the original file is changed. The info dictionary of the notification can contain `kMDQueryUpdateChangedItems` and `kMDQueryUpdateRemovedItems` keys.

For performance reasons added results are not indicated in progress notifications, to avoid the cost of creating the result objects.

This notification is only sent to the application's notification center.

Notification Info Keys

Query Result Change Keys

Specify the items that have changed in the query results.

```
const CFStringRef kMDQueryUpdateAddedItems;
const CFStringRef kMDQueryUpdateChangedItems;
const CFStringRef kMDQueryUpdateRemovedItems;
```

Constants

`kMDQueryUpdateAddedItems`

An array that identifies the items that have been added to the query results. This list only contains result objects that have previously been created, result objects that have not been created are not included.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

`kMDQueryUpdateChangedItems`

An array that identifies the items that have changed in the query results. This list only contains result objects that have previously been created, result objects that have not been created are not included.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

`kMDQueryUpdateRemovedItems`

An array that identifies the items that have been removed from the query results. This list only contains result objects that have previously been created, result objects that have not been created are not included.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

Query Search Scope Keys

Specify the scope of a query's search.

```
const CFStringRef kMDQueryScopeHome;
const CFStringRef kMDQueryScopeComputer;
const CFStringRef kMDQueryScopeNetwork;
```

Constants

`kMDQueryScopeHome`

Specifies that the query should be restricted to the volume and directory that contains the current user's home directory.

Available in Mac OS X v10.4 and later.

Declared in `MDQuery.h`.

`kMDQueryScopeComputer`
: Specifies that the query should be restricted to all locally mounted volumes, plus the user's home directory (which may be on a remote volume).

  Available in Mac OS X v10.4 and later.

  Declared in `MDQuery.h`.

`kMDQueryScopeNetwork`
: Specifies that the query should include all user mounted remote volumes.

  Available in Mac OS X v10.4 and later.

  Declared in `MDQuery.h`.

Discussion
These constants can be passed in the *`scopeDirectories`* array to the function `MDQuerySetSearchScope`.

Result Relevance Sorting Key

Key used in a user notification's description dictionary that indicates the relevance of a result.

```
const CFStringRef kMDQueryResultContentRelevance;
```

Constants
`kMDQueryResultContentRelevance`
: A CFNumberRef with a floating point value between 0.0 and 1.0 inclusive.

  Available in Mac OS X v10.4 and later.

  Declared in `MDQuery.h`.

Discussion
The relevance value indicates the relevance of the content of a result object. The relevance is computed based on the value of the result itself, not on its relevance to the other results returned by the query.

The relevance value is for the content of the object only, not on the result item as a whole, and may not be computed if the item matches theq uery through evaluation of other attributes

If the value is not computed it is treated as an attribute on the item that does not exist.

Document Revision History

This table describes the changes to *MDQuery Reference*.

Date	Notes
2005-06-04	Added a note that progress notifications are sent to the application's local notification center.
2005-04-29	Added new query API. First public version.
2004-06-28	New document that describes the opaque type for creating and executing metadata queries.

Document Revision History

Index

K

M

Q

R