Infrastructure - Task #1937

Story # 2000 (Closed): Edits for API docs and implementations for consistency

MNRead.listObjects() - Naming of startTime and endTime params.

2011-10-27 02:16 - Roger Dahl

Status:	Closed	**Start date:**	2011-10-27
Priority:	High	**Due date:**	
Assignee:	Dave Vieglais	**% Done:**	100%
Category:	Documentation	**Estimated time:**	0.00 hour
Target version:	Sprint-2011.49-Block.6		
Milestone:	CCI-1.0.0	**Story Points:**	
Product Version:	*		

Description

We should have the date time span parameters be named the same across all the calls. listObjects() uses startTime and endTime. I prefer the fromDate and toDate parameters used in getLogRecords(). Alternatively I think start/stop or begin/end is better than start/end.

History

#1 - 2011-11-03 20:15 - Dave Vieglais

- Status changed from New to In Progress

- Priority changed from Normal to High

agree with fromDate and toDate, but this change would affect a number of operations.

Requires discussion to determine if it's worth fixing.

#2 - 2011-11-04 19:15 - Dave Vieglais

- Parent task changed from #1906 to #2000

#3 - 2011-11-04 19:18 - Dave Vieglais

The general recommendation is as follows:

1. Use fromDate and toDate as the beginning and end time fields respectively.
2. Use fromDate <= date < toDate instead of fromDate <= date <= toDate.

Using less-than for the top end of the range makes paging with date slices simpler, since the toDate can be used as the fromDate for the next page with no changes.

#4 - 2011-12-12 23:35 - Dave Vieglais

- Status changed from In Progress to Closed

Docs updated, commit revision #6166

#5 - 2011-12-13 13:02 - Dave Vieglais

Logic for use of fromDate <= date < toDate:

On member nodes the primary uses of listObjects and getLogRecords is for retrieval of those lists for further processing, and the time range constraints provide a mechanism for slicing the list of available records.

Say we have three cases for comparison:

1. fromDate < date <= toDate
    1. fromDate <= date <= toDate
    2. fromDate <= date < toDate

If we have a set of records with actual time of event in column 2, and the recorded time (stored at the machine or software resolution, such as 1 ms) in column 3:

```
1  0.00  0.00
2  0.25  0.00
3  0.50  0.00
4  0.75  0.00
5  1.00  1.00
6  1.25  1.00
7  1.50  1.00
```

Say listObjects is called at 0.75, which means the query is made with a fromDate value of "0.00". In case 1, records 5+ would be returned; in cases 2 and 3 records 1+ would be returned. Hence records 1-4 would be excluded for case 1.

Continuing the example, perhaps the next time listObjects is called the total record list looks like:

```
1  0.00  0.00
2  0.25  0.00
3  0.50  0.00
4  0.75  0.00
5  1.00  1.00
6  1.25  1.00
7  1.50  1.00
8  1.75  1.00
9  2.00  2.00
10 2.25  2.00
11 2.50  2.00
```

The last record retrieved from the previous call was #7, which has a recorded time stamp of "1.00", so the list objects query would have fromDate = 1.00

In case 1, records 9+ would be returned; for cases 2 and 3, records 5+ would be returned.

So case 1 misses #8. Cases 2&3 include 5,6,7 even though they were retrieved previously.

Since detecting a known error (duplicate) is a bit simpler than an unknown error (missing records), the conclusion is that the inclusive lower range should be used and that the algorithm for listObjects and getLogRecords needs to be modified to detect and gracefully handle duplicates.

As indicated by Rob, conventions generally follow the range query specified like fromDate <= date < toDate, hence listObjects and getLogRecords should use that comparison for practical reasons and to follow convention (which likely arose from exactly this type of scenario - time moves forward).