Overview of Iterator Usage

This document contains an overview of how iterators are used.

Iterators are used in many operations to allow the NMS to deal more easily with retrievals which may return a very large amount of data. They provide a mechanism to retrieve data batch by batch - the ideal size of a batch is specified by the NMS.

In a typical case, an NMS requests a list of objects. With this request it can specify the maximum number of objects it can handle in the first reply, the number sent by the EMS may be less than this. The EMS gets a snapshot of the current objects to be returned: the response will contain a list with the specified number of objects and a reference to an iterator which can subsequently be used by the NMS to access further objects of the snapshot.

The generalised form of such an operation is:

```
getAllObjectName(
      (in qualifiers) /* some operations may supply qualifiers
      */
      in unsigned long how_many,
      out objectList list,
      out iterator iteratorReference)
raises(globaldefs::ProcessingFailureException);
```

The `how_many` parameter determines the maximum number of response entries there will be in the `list` output parameter. The `iteratorReference` provides access to the remaining objects, if any.

If the `list` contains the complete set of *`ObjectName`* objects then the `iteratorReference` will be a reference to a `CORBA::Object::_nil` object. If there are more, `list` will contain the first batch of objects known to the EMS and the `iteratorReference` will provide access to further objects. If 0 is supplied for `how_many`, no objects will be returned in the `list`, and all objects will have to be retrieved from `iteratorReference`.

The EML may return less than `how_many` entries if, either less than `how_many` entries exist, or if the EML determines that `how_many` exceeds the server's stated performance restrictions. For example the EML may have 100 objects to return, the NML may request 50 via `how_many` and the EML may return 20 along with an `iteratorReference`.

The `iteratorReference` parameter is used by the NMS to request further batches. This reference could be used to query how many objects could be returned by the operation (`getLength`) and subsequently the NMS could start retrieval or destroy the iterator. If all objects were contained in the `list` then this will be a reference to a `CORBA::Object::_nil` object. Each entry will be returned at most once; hence consecutive calls to `next_n` can be used to retrieve all of the additional entries in the selected set. Again the EML may return less than `how_many` entries if, either less than

how_many entries exist, or if the EML determines that how_many exceeds the server's stated performance restrictions.

In some cases, it may not be feasible for the EML to obtain the total number of elements that will be returned by the iterator:

- The EML may have to first fetch all of the data that has been requested (making a local copy) and then count the number of elements being iterated over. This could prove to be particularly inefficient.
- The EML may be able to obtain a count of the number of elements, but due to concurrent modifications, the actual number may vary over the time of the iteration.

A client of the interface can always determine when all data has been collected by repeatedly fetching chunks of data until the iterator indicates that no further data remains.

In the cases where the EML is not able to suitably determine the length the getLength method should respond with the EXCPT_CAPACITY_EXCEEDED exception.

The EXCPT_TOO_MANY_OPEN_ITERATORS exception will be returned if the EMS has reached an implementation limitation. A minimum of **10 iterators** shall be supported by an EMS.

In this interface the iterators have the following operations:

getLength: returns the total number of elements in the iterator (not the remaining elements still to be traversed !!). Example: If the EML has 100 objects, and 10 are returned in the initial list, the getLength method should return 90, regardless of how many objects were or where not retrieved with the next_n method. The EXCPT_CAPACITY_EXCEEDED exception will be returned if the EMS cannot efficiently provide a value for the number of elements on this occasion. The operation next_n may still be performed.

next_n: returns at most n entries; it may return fewer. Returns true if there are more entries to be returned; otherwise it returns false, destroys the iterator and releases the memory being used. The EML may choose to limit the number of objects returned in a single request to prevent performance issues.

destroy: If client decides not to access the remaining objects, it can also invoke the "destroy" operation to delete the iterator object.

Revision History

Version	Date	Description of Change
3.0	November 2006	Conversion of iterators into new template

Acknowledgements

<FirstName> <LastName> <Company>

How to comment on the document

Comments and requests for information must be in written form and addressed to the contact identified below:

Keith Dorking Ciena

Phone: +1 678 867 5007

Fax: +1 678 867 5010

e-mail: kdorking@ciena.com

Please be specific, since your comments will be dealt with by the team evaluating numerous inputs and trying to produce a single text. Thus we appreciate significant specific input. We are looking for more input than wordsmith" items, however editing and structural help are greatly appreciated where better clarity is the result.