

1.0 reducingHandlerBuilder.build() - create a new instance of ReducingHandler class, the call invoke creation of several mandatory objects(1.1-1.4), create and return the instance of ReducingHandler(1.5-1.7)

2.0 void accumulateReducingData(unsigned long long itemId,

 Poco::SharedPtr<reduce\_types::ReducingInputMessage>& reducingInputMessage) – takes as parameters reducingInputMessage (see Reducer functional object message protocol.doc) and an itemId to which the input data belong to; invoke(2.1) to check if such itemId is already exist. If no – create necessary structures(3.0-3.2). Increments count of accumulated data items(4.0) and save data in the SphinxTaskReducerManager(4.1). If during (4.1) was arised WrongJSONStructureException exception then involves the (5.0) to notification the error

5.1makeReducing(unsigned long long itemId) - execute reducing task for data related to itemId, return result as ASM::reduce\_types::ReducingOutputMessage(native language format)
If itemId is valid – (5.2) returns a SphinxResult object in json format and executes (6.0) to build a ReducingOutputMessage in native language format
else runs (6.3) and (6.4) to build ReducingOutputMessage object

7.0 deleteReducingTaskBy(unsigned long long itemId) – remove all data structures related with input itemId. If the itemId is valid – execute (8.0-8.1), else (9.0)

9.1. getAccumulateReducingPartsNumberBy(unsigned long long itemId) – return number of data item parts accumulated for the itemId, invoke (9.2)

10.0 unsigned int getTasksNumber() - return the number of unique itemIds which have data in the reduceHandler's structures, invokes 10.1

11.0 getExceededTTLTasks( unsigned int maxTaskNumber, std::vector<unsigned long long>& exceededTTLTasks, unsigned int ttl = 0 ) - add to the vector(exceededTTLTasks) itemId which ttl value is less that input ttl parameters; adds to the vector no more that maxTaskNumber elements; invoke(11.1)

12.0 int cleanupExpiredTasksByTTL(unsigned int maxRemoveTasks = 1, unsigned int ttl = 0) – removes elements from the global queue of running tasks number and inserts them in to the global queue of “terminated by TTL tasks”, removes no more that maxRemoveTasks elements which ttl value is less that input ttl parameters; returns number of removed elements, invokes \_reducingHandler->cleanupExpiredTasksByTTL(12.1)

13.0 unsigned int cleanupExpiredTasksByTTLQueue( unsigned int maxRemoveItems, unsigned int minTerminateTime ) - remove elements from the global queue of “terminated by TTL tasks”, removes no more that maxRemoveTasks elements which ttl value is less that input minTerminateTime parameters; returns number of removed elements; invoke (13.1)

14.0 bool isTaskExpired( unsigned long long taskId, unsigned int ttl = 0 ) - check if the taskId ttl value is less that input ttl parameter(14.1).

15.0 bool isTaskTerminatedByTTL( unsigned long long taskId ) - check if the taksId was finished by ttl condition(15.1)