Grouper Messaging System - full sync messages

Currently, Grouper sends messages (via AMQP or Amazon SNS) regarding changes going on in Grouper with create/read/update/delete messages pertaining to groups, privileges, memberships, attribute changes, and possibly other activities. With respect to memberships in particular, the change and resulting message is incremental and includes essential information about the change along with the subject, the group and the action - usually ADD /DELETE. While there appears to be general agreement Identity Management related messages need to be delivered in order and never deleted (a deleted message implies not delivering in order), it remains possible actions represented by messages "get lost" either by the message broker system or by the consumer/processor itself on the receiving end of Grouper messages, OR an application fails in some way and needs to be synchronized with data /groups from Grouper.

There are various integration patterns for applications and the following methods should address the scenarios for handling the overall problem.

1. **Sync and Call-back method**: send a "sync-and-callback" message to the consumer/application and then it turns around and communicates with Grouper or LDAP or other systems to synchronize data. Grouper Web Services could be used or if Grouper is provisioning to LDAP then reaching out to LDAP might be viable as well. Implied here is not only having a message consumer reading from a queue (usually with appropriate access control to the queue) but also includes configuring Grouper or LDAP or whatever other component to have access control configured as well for the case of the "sync-and-callback" message. A sync message could be identified by the consumer in a variety of ways:
    a. additional field in the message indicating a "sync-and-callback" message
    b. the absence of a subject in the incremental message implies 1-(a)
2. **FULLSYNC-message-fragments method:** send a "FULLSYNC fragment" message to consumer with additional information indicating which fragment for the object being sent the message belongs. Provide sufficient information so the consumer can put the multiple fragments back together should they arrive out of order. Provide configuration to control size of message or number of subjects in the message.
3. **FULLSYNC-single-message method**: send a "FULLSYNC" message to the consumer/application looking like an incremental change message for membership and populating the subject array (normally only having 1 item) with all the full membership of the group. Could be implemented as 2 above and specifying no limit to a fragment - size limit = 0

As an "application/consumer" might be configured to receive many groups or a portion of the Grouper tree assigned to the app it would be needed for the mechanism to traverse the Grouper Stem tree from a starting point Stem thereby making it easy to sync up many groups or even all groups if Root is the starting point.

Initially, only interested in memberships, but there may be a desire to sync up privileges and other attributes or objects in the future.

There are many applications and scenarios in which a variety of integration techniques and security concerns may be at play causing the 3 integration patterns noted above to be applied.

Execution should be handled by Grouper Loader OTHER jobs making use of Quartz cron and supplying the needed parameters to control the jobs. There may be many jobs configured with different parameters. Parameters based on above include: limit_number_of_subjects, limit_size_of_message, stem_to_sync, message_type (sync-callback, fullsync) or it's all fullsync with the parameters governing behavior - a fullsync with no subjects implies callback.