SPECIFICATION OF THE NGAS CLIENT API AND ARCHIVE MULTIPLEXING/PROXY MODE:

=API Server Multiplexing:

A new function will be added to the C-API setServerList() and a new method will be added to ngamsPClient, ngamsPClient.setServerList() with which it is possible to specify a list of servers + associated ports.

Internally before sending a command, the APIs will take the next server/port set specified (cyclically) and use that for the communication. If a communication error is reported, the next server/port set in the list will be used. If all fail, the functions/method will return an error to the host application. The last server/port set used, will be remembered.

Having specified a list of servers/ports, it is possible to omit the host/port parameters to the various functions/methods of the APIs (their values will be ignored).

For the command line utilities, a new command line parameter will be provided: "-serverList <Host>:<Port>,<Host>:<Port>,...", which internally calls the serServerList() function/method. Having specified "-serverList" it is not enforced to specify a "-port" to the command line utilities.

Note, if the ticket suggesting a sinplication of the APIs/command line tools is implemented (DFS01570), the multiplexing and DFS01570 will be implemented together.

=Archive Proxy Mode:

For the MNUs, a high-level Stream definition is used as proposed in ticket DFS01668, whereby Target Hosts are specified for a given Stream, rather than Storage Sets, e.g.:

```
<Streams>
  <Stream  MimeType="image/x-fits">
    <Host Id="ngau1"/>
    <Host Id="ngau2"/>
  </Stream>
  ...
</Streams>
```

The Stream definition for the actual Archiving Units remains the same as now. The Storage Set definition can be omitted for the NMUs.

A new method in ngamsDb will be provided, to allow to query the next Target Archiving Unit from the DB (ngamsDb.findTargetHost()). The NMU will multiplex between the various NMU's specified to spread the load among these.

The configuration parameter Server.ProxyMode will be used as for the RETRIEVE Command:

1. ProxyMode=1: Act as proxy for the request. Find the most suitable Target NAU, ping this (send STATUS Command). If OK, forward the Archive Request to the chosen node. After the handling in the Target NAU has ended, the response from the NAU is returned to the client.

2. ProxyMode=0: Find the most suitable node, ping this (send STATUS Command) and return an HTTP Redirection Response to the client. The client must then itself, re-submit the request to the given host. To implement this efficiently, a new option could be provided for the STATUS Command: "-getArchivingUnit". In this case, the data is not submitted by the client with an Archive Request before the alternative NAU has been returned by the server. If ProxyMode is 0 and an Archive Request is received, an error is returned by the contacted node. A new function/method could be provided in the C/Python APIs, ArchiveMultiplex(), which internally handles the negociation of the Target NAU and the subsequent archiving of the data.

If a communication problem is encoutered by a contacted server while this pings a candidate NAU, the next one in the list will be contacted. If none of the NAU's specified can be contacted, an error will be returned.

In the first implementation, it will not be possible to let a node act as Archive Proxy Server and NAU at the same time. This could be added later if relevant. This option seems not relevant however, since if it is desirable to let the clients multiplex between NAUs directly, they can use the multiplex feature of the APIs/clients.

An Archive Proxy Server only accepts Archive Requests, if Server.AllowArchiveRequest is 1.

EOF