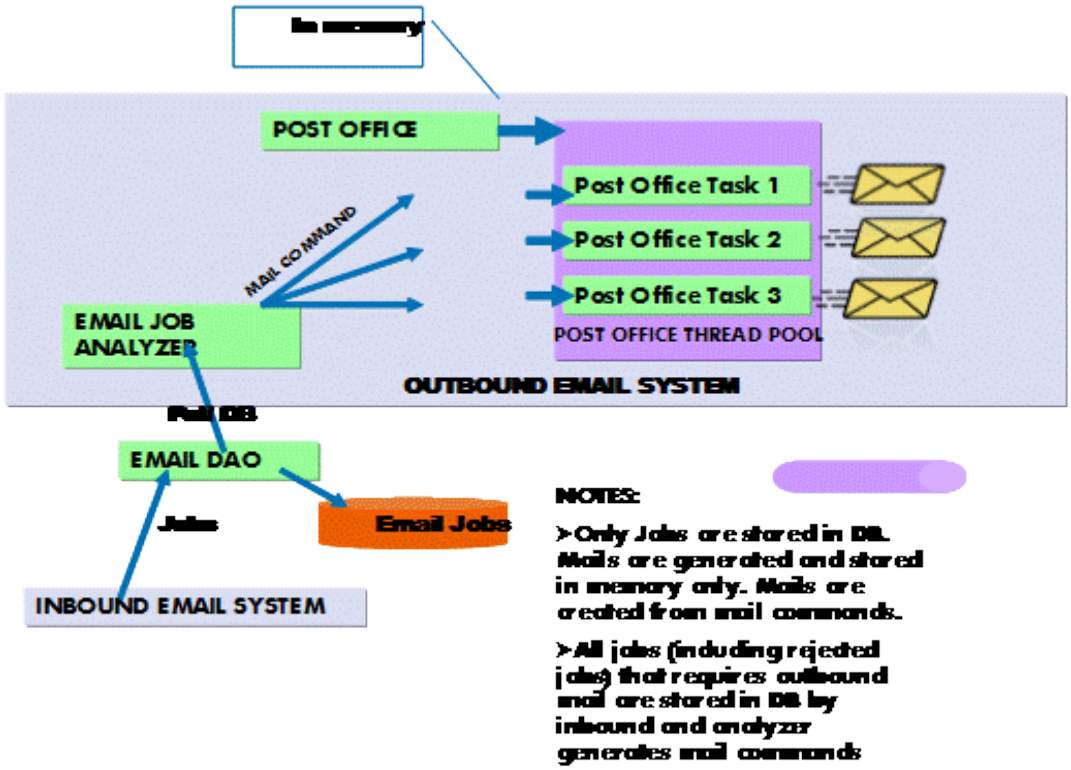
Email Outbound Design Notes

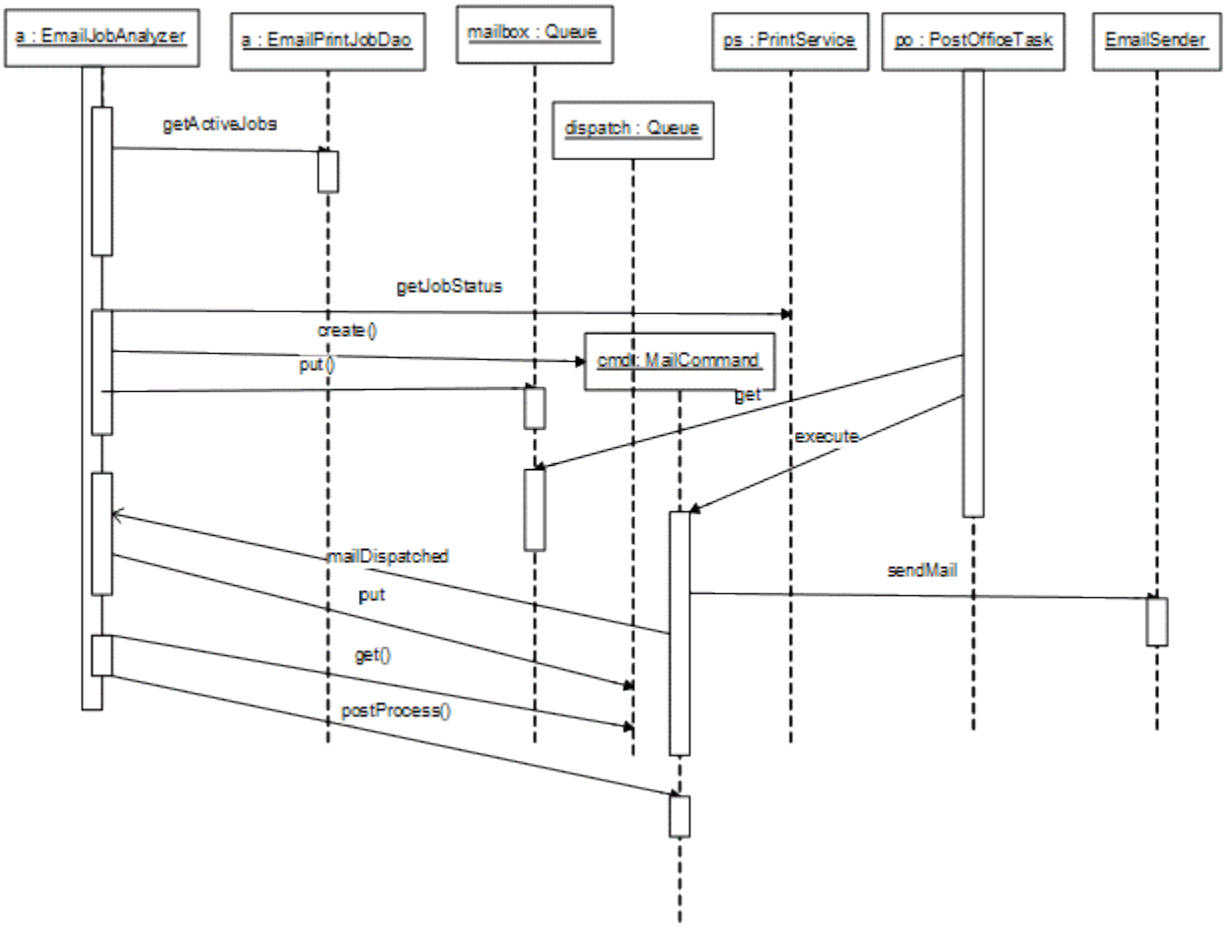


Job Analyzer is responsible for analyzing job (currently by polling DB) and generate mail commands. There is one Job Analyzer thread per shard allowing the system to scale across shards and concurrently execute tasks from multiple DB shards.

The mail commands follow a command pattern and encapsulate the construction and sending of the following mails:

2. Generic Ack
3. Generic Failure
4. Delayed
4. Print Completed
5. Print Error
6. Other Mail (Mail command delegate for all email sent by other components of ePrint)

The following sequence diagram depicts the main flow of the email outbound component.



The Job Analyzer (EmailJobAnalyzer.java) first inspects the database for any new email jobs. If found and if the email print job has been accepted for submission, the analyzer will use the print service API to get the status of the jobs. Based on the status seen, this logic will create an appropriate mail command that is applicable to the current job status and puts the mail command into a mailbox. Each Job analyzer thread posts its mail commands to a different mailbox. The mailbox is a queue object. The Post Office (PostOffice.java), once opened will manage a thread pool of post office tasks (PostOfficeTask.java), each task looking into different mailboxes for incoming mail command to dispatch. Once a mail command arrives the post office task immediately executes the specific mail command which includes construction of the email and sending the email out to the recipient address and notifying the analyzer that the mail was successfully dispatched. Once analyzer knows that the mail has been dispatched, it does a postProcess on the mailcommand and destroys the mail command.

Each Job Analyzer has a main loop that wakes up periodically and inspects the respective job DB shard for any incoming email jobs that is not yet processed. It also actively monitor

the in flight jobs using the print service's (PJP) job status api and sent outgoing mail accordingly.

As seen above, the MailCommand Interface and the concrete mail commands follows the command pattern and is designed as such in order to ensure proper encapsulation of mail commands and better maintainability.