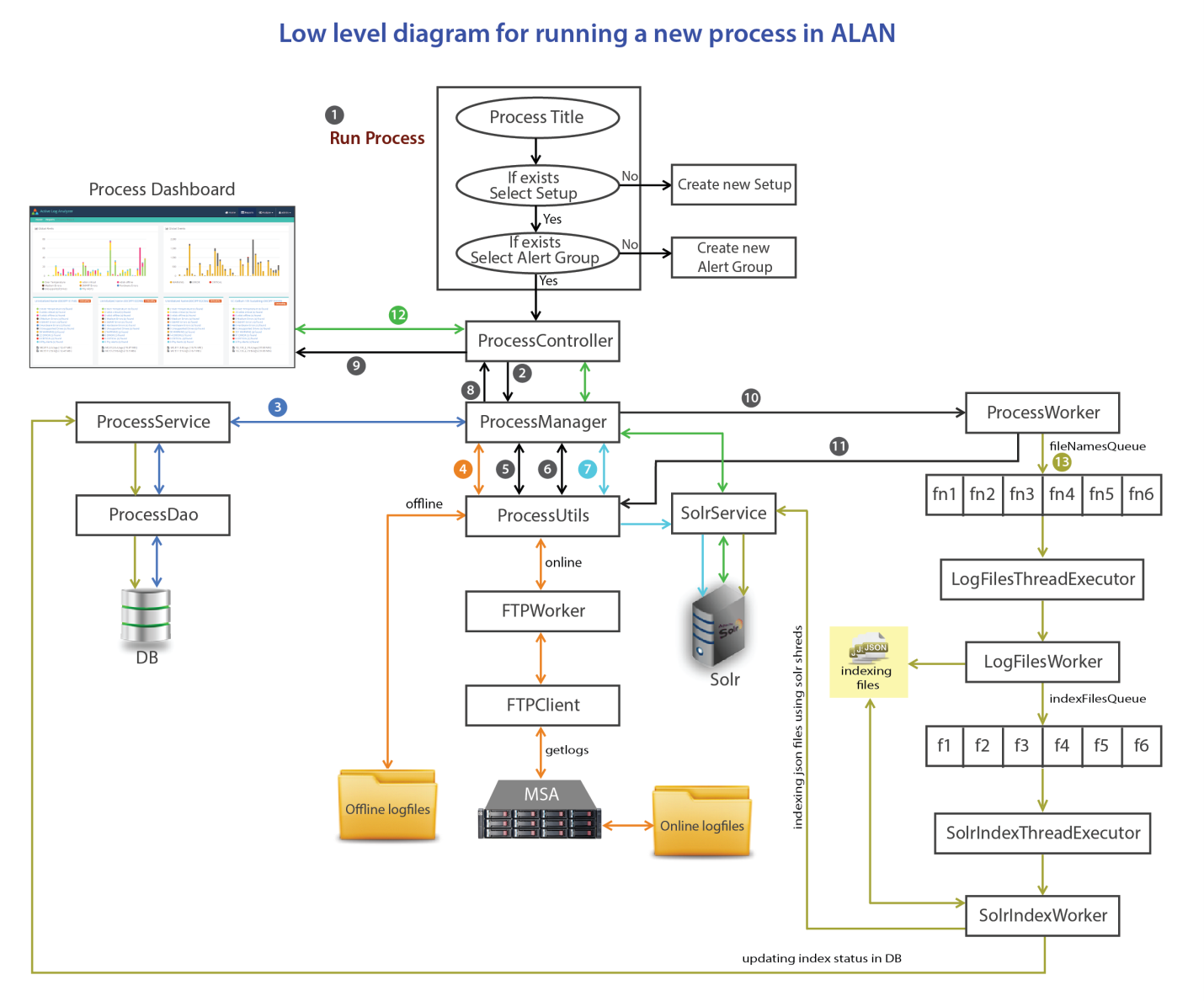
The following ‘Functional Flow’ diagram depicts how a log is processed in ALAN:



The 13 (thirteen) stages involved in the running a process are explained below:

1. In this stage, you need to open the ‘Run’ page from the ALAn application and provide the required fields such as process title, setup, alert group and, click the ‘**Run Process**’ button.
2. In second stage, the request is passed to the ‘**ProcessController**’ using the ‘**RequestMapping**’ annotation and then it is forwarded to the ‘**ProcessManager**’.
3. In next stage, the ‘**ProcessManager**’ creates a new process record in the DB (Database) using the ‘processService’ object and gets the ‘processId’.
4. The ‘**ProcessManager**’ requests ‘ProcessUtils’ to get the log files location as below:
   * If the provided setup mode is *Offline* then ‘ProcessUtils’ returns the log files location from the setup object (For *Offline* setups, user has to provide the log files location when creating the *Setup*).
   * If the setup mode is *Online*, then ‘ProcessUtils’ forwards the request to the ‘FTPWorker’ that runs the ‘GETLOGS’ command using ‘FTPClient API to download the online logfiles from the specified MSAs to a temp location (as specified in the *Properties* file).
5. The ‘**ProcessManager**’ then requests ProcessUtils to get the log files from the ‘path.ProcessUtils’ extracted zip files. Only *.logs* files are exctrated and a map containing midplane and its associated log file names are returned.
6. In the 6th stage the ‘**ProcessManager**’ requests ‘ProcessUtils’ to process the first midplane in the returned map. The ‘ProcessUtils’ generates and returns a report in the form of the Result object.
7. Then, the ‘**ProcessManager**’ requests ‘ProcessUtils’ to save the reports in *solr*.
8. The returned report is sent to the ‘ProcessController’ as response.
9. The ‘ProcessController’ sends response to the process dashboard.
10. If more than one midplane exists in the map returned in 5th stage, then the ‘**ProcessManager**’ assigns the job of processing remaining storage arrays to the ‘ProcessWorker’.
11. In this stage, the ‘ProcessWorker’ executes the stages 6th and 7th repeatedly; untill all the Midplane reports are saved in the *solr*.
12. Meanwhile, the process dashboard continue sending the asynchronous(AJAX) requests to the ‘ProcessController’, to get the completed dashboard reports. The ‘ProcessController’ sends the response to the process dashboard by requesting ‘ProcessManager’ which is in communication with *solr.*
13. After preparing all the dashboard reports, the ‘ProcessWorker’ invokes ‘BlockingQueue(fileNamesQueue)’ by adding the ‘FileNames’ objects to it. The ‘LogFilesThreadExecutor’ invokes and assigns the added ‘FileNames’ objects to the ‘LogFilesWorker’ to create indexing files in *json* format in a temporary location (as specified in the *Properties* file). As soon as indexing files creation starts, the ‘SolrIndexThreadExecutor’ invokes and assigns them to the ‘SolrIndexWorker’ which then runs the ‘post.jar solr’ command to commit the ‘json’ files in *solr*. After committing all the ‘json’ files in *solr*, the ‘SolrIndexWorker’ updates the index status in database.