**Valid trigger for jsondump.py**

A valid trigger for the python script jsondump.py is an **event log trigger**. An Event log trigger is watching the event logs for changes based on certain conditions appearing in the logs eg:

* A specific/a group of event IDs ie. *The support team would like to monitor event ID 855,908,1023*
* A specific event type. There are 4 event types:

1. "Error" - An error event, indicates a significant problem the user should be aware of; usually a loss of functionality or data
2. "Warning"- A warning event, indicates a problem that must not be taken care of immediately, but may indicate conditions that could cause future problems
3. "Information" - An information event, indicates a significant and successful operation
4. "SuccessAudit"- A success audit event, indicates a security event that occurs when an audited access attempt is successful; for example, logging on successfully
5. "FailureAudit"- A failure audit event, indicates a security event that occurs when an audited access attempt fails; for example, a failed attempt to open a file

A time frame can also be set such that if , a specified event/ a condition found in the logs occurs for a specific *n* number of times, it will trigger an alert to the support team as well as trigger jsondump.py to scrape the event logs and migrate them into the ELK stack.

The implementation of this event log trigger would be feasible because system simply has to scan the content of the logs for the conditions set for a specific timeframe, and trigger the python script to run and pump the data into our ELK stack if the condition is met.

With this event trigger, there will still be an occurrence lag between an event occurrence and reporting but it will be significantly be less than a time-driven trigger. A time driven trigger is when a specific time is specified when the python script would be run. With a time driven trigger, we might have to wait till the end of day to monitor our logs from our ELK stack. However, performance of a time driven trigger, would be better than an event trigger as it does not require the system to monitor the logs 24/7 as it has been set to only run the python script at a specific time of the day/week/month.

In conclusion, there is no straightforward answer to which is the best trigger to initiate the scraping of logs as it depends entirely on why we need to monitor our logs and what exactly are we interested in finding. For instance, during development work, we might be interested in looking at our logs, to identify bugs during daily deployment. So how often we need to analyze our event logs depends on how often we make changes to our applications/ perform deployments. Hence, a time driven trigger would be useful *ie trigger* ***jsondump.py*** *at the end of each deployment day*. Another scenario can be also based on support requests to analyze logs. Support teams regularly have to monitor logs to answer queries regarding failed logins/site issues and hence, a manual trigger based on per requests can also be helpful for the support team to do logs monitoring.