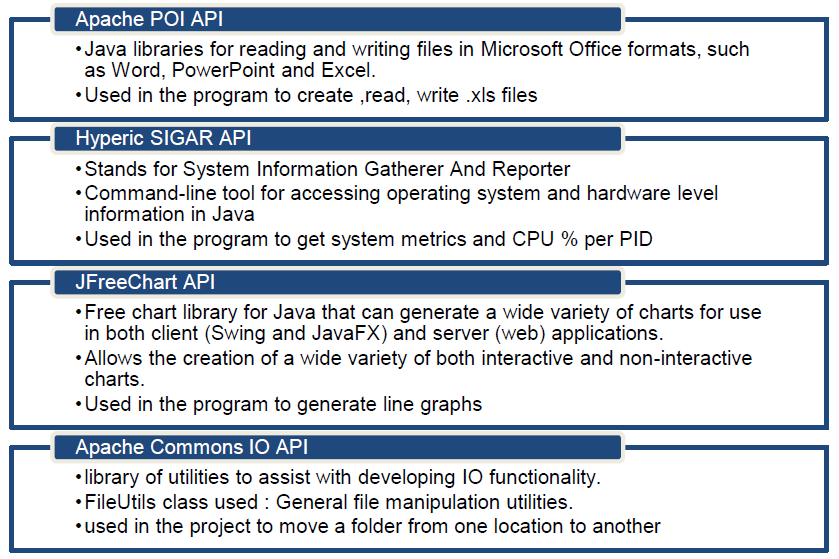
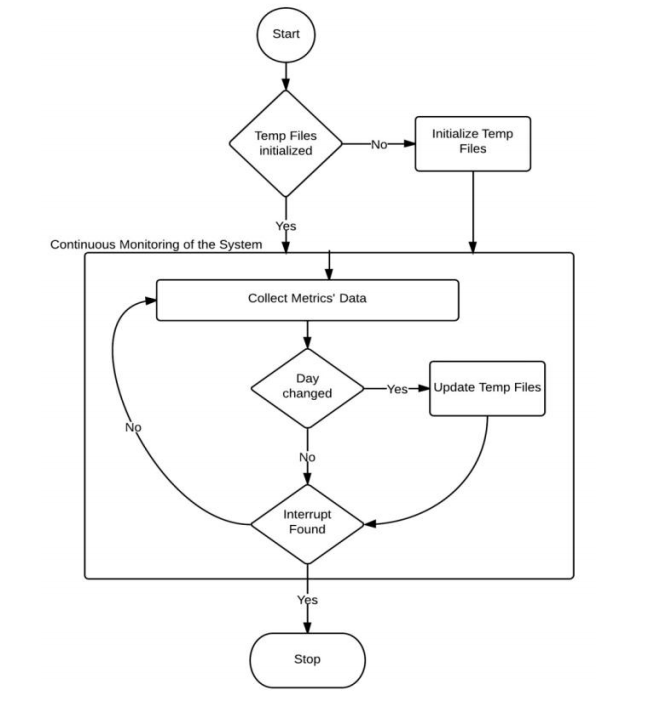
Developer Manual  
Performance Dashboard for Data Services and BO Server  
**Table of Contents**  
Introduction .................................................................................................................1  
System Overview .......................................................................................................1  
System Metrics......................................................................................................1  
BODS Specific Metric ..........................................................................................1  
BO Specific Metric................................................................................................1  
Data Sources Used .....................................................................................................2  
APIs Included in the Project.....................................................................................3  
Backend Development.............................................................................................3  
Frontend Development ...........................................................................................7  
Index of Java Files in the Zip File ........................................................................9  
How to Create a Jar File........................................................................................10  
Limitations.................................................................................................................10  
Future Work..............................................................................................................11

**Introduction**  
The purpose of this booklet is to help developers understand the tool – its functionalities,  
algorithms, limitations and future work that can be done with it.  
**System Overview**  
This software is a Java-based platform-independent plug-and-play tool that obtains  
information and generates a performance dashboard with respect to the following metrics:

**System Metrics**  
•CPU Utilization %  
•RAM Utilization %  
•Disk Space Utilization %  
•Disk Read/Write Bytes (KB/sec)  
**BODS Specific Metrics**  
•CPU Utilization % per BODS Job  
•Disk Read/Write Bytes per BODS Job  
**BO Specific Metrics**  
•Number of Users Logged In to BO Server per Day  
•Number of Connections to BO Server per Day  
•Number of User Sessions in BO Server per Day  
•Number of BO Reports Refreshed per Day  
•Size of BO Input and Output FRS  
NOTE: Disk Read/Write Bytes per BODS job and BO specific metrics have not been  
implemented yet

**Data Sources Used**  
•**System Metrics**  
The system metrics are obtained through the Hyperic SIGAR API which is included in  
the project. SIGAR Stands for System Information Gatherer And Reporter which is a  
command-line tool for accessing operating system and hardware level information in  
Java  
•**BODS Specific Metrics**  
•CPU Utilization % per BODS Job  
1. Whenever a job is run, its corresponding trace file is generated in  
the log folder present in the BODS sever.  
2. The trace file contains the name of the job and the dataflows that  
are part of that job and also the PIDs (Process Ids) of each data  
flow  
3. All the PIDs of running dataflows are obtained and their  
corresponding CPU Utilization % is recorded using a SIGAR  
function.  
4. CPU utilization % of the job at any point is the sum of the CPU  
Utilization % of all the dataflows of that job running at that time.  
  
**APIs Included in the Project**

  
**Backend Development**  
There are 2 aspects to the software:  
1. **The Monitor** which collects the data of the selected metrics and stores it in .xls files  
2. **The Dashboard** which reads the data from the .xls files and plots the graphs  
**Algorithm for the Monitor**  
1. Selected metrics are read  
2. Settings fields are read  
3. Start button is pressed  
4. if (the data files don’t exist)  
a. Initialize the files  
5. Do (every 20 seconds)  
a. If (new day)  
i. Calculate minimum, maximum and average values for  
previous day’s data and output in final excel file for that  
metric  
b. Collect selected metric’s data and output in temporary file for  
that metric  
c. If (stop button was pressed)  
i. Go to Step 7  
6. Go to Step 5  
7. Exit  
**Flowchart for the Monitor**  
  
**Algorithm for collecting CPU Utilization by BODS jobs (**CpuByBodsJobs.java**)**  
1. Do (for 30 days)  
a. If (new day)  
i. Monitor BODS Logging folder every 20 seconds  
ii. If (new trace file created)  
1. Get the name of the Job  
2. If (Workbook sheet already not created for job)  
a. Create Workbook sheet with the name as  
the Job name and initialize required  
dummy row and column names  
b. Add job in active jobs list  
3. If (20 seconds have passed)  
a. Iterate through all jobs in active jobs list  
b. Open workbook sheet and trace file for  
each active job  
c. For (all dataflows in the sheet)  
i. If (dataflow has ended)  
  
1. Remove corresponding  
record  
ii. If (new dataflow started and not  
ended)  
1. Add a record for it  
2. Get CPU Utilization for each  
d. Add CPU Utilization for records  
e. Update min/max/avg values and  
corresponding time stamps if need be  
f. If (job has ended)  
i. Remove all dataflows from  
corresponding sheet of the  
temporary data workbook  
ii. Remove job name from active jobs  
list  
4. Add records in the final file sheet  
5. Reset min, max, avg timestamps of each .xls sheet  
of all jobs  
6. Update the date variable to reflect the current  
date.  
7. dayctr++  
8. If (dayctr > 30)  
a. Break from monitoring loop and exit  
program  
**Frontend Development**  
Java Swings have been used to create the GUI. The Drag-and-Drop GUI Builder  
functionality has been used in Netbeans IDE to design the panels in the GUI.  
**Main Components**  
The main panels of the front end are:  
•**The Monitor**  
This screen opens up whenever the application is started so that the  
user can set the paths for the required files; choose which metrics are  
to be monitored and to start/stop the monitoring. A similar panel  
opens up to generate the dashboard, wherein users are given the  
option of choosing which all metrics they want to view. Whenever the  
Monitor Window is closed, the application continues to run and the  
screen minimized to an icon in the System Tray. From this icon, we can  
reopen the Monitor screen or stop the execution of the program  
altogether.  
•**The Chart Panel**  
This screen contains the graphs of all the metrics previously selected  
by the user. The graphs generated display the metric name as well as  
the legend describing what is the significance of each line present in  
the graph. On hovering the mouse over any data point, a tooltip  
appears with the date, time and the data value corresponding to that  
point. The graph can also be zoomed by dragging the mouse and  
selecting the desired portion and it can be zoomed out by dragging the  
mouse from right to left. The user gets the option of modifying the look  
of the graphs and can also save the graphs in various formats by right clicking on the graph.  
•**User View Options**  
Each graph being displayed on the dashboard has an “Options” button  
associated with it. On clicking, a screen asks for user’s preferences in  
terms of what information he or she wants to see, allows him/her to  
specify the number of days that he wants to observe data and the  
threshold value in order to compare the difference between the two  
values with ease.  
In each panel, the action listeners of all components has to be appropriately  
modified in order to achieve the desired result in terms of actions performed  
on selecting/ unselecting/ clicking them.  
  
**Index of Java Files in the Zip File**  
**File Name Contains Code for**  
CpuByBodsJobs Monitoring CPU Utilization % by BODS Jobs  
CPUJObsOptions User view options panel for graph of CPU Utilization %  
by BODS Jobs  
CPUOptions User view options panel for graph of CPU Utilization %  
Dashboard Dashboard menu as well as the final chart panel  
generated after it  
DiskOptions User view options panel for graph of Disk Space  
Utilization %  
IoOptions User view options panel for graph of Disk Read/Write  
Bytes  
Monitor Monitor menu  
PerformanceMonitor The main project file from where the entire application  
is initiated  
RAMOptions User view options panel for graph of RAM Utilization  
%  
Settings Settings panel where user can specify the required  
paths  
StartMonitor Initiates the monitoring of selected metrics  
SystemCpu Monitoring CPU Utilization %  
SystemDisk Monitoring Disk Space Utilization %  
SystemIO Monitoring Disk Read/Write Bytes of the System  
SystemRam Monitoring RAM Utilization %  
  
**How to Create a Jar File**  
Refer to the following tutorial: http://youtu.be/WTfjbOIfi10  
**Limitations of the Application**  
1. If the monitor is closed, all the Path Settings will be lost and in the next  
run of the application, if a new path is entered then new data files will  
be created leading to disconnected sets of data  
2. If Stop button is pressed or the application is terminated for more than  
one day, the data in the temporary files from the new day will keep  
getting added on to the data of the day that the program was  
terminated.  
3. Design (look only) of the CPU by BODS Jobs and Disk Space Options has  
to be reworked.  
4. If the monitor is stopped and restarted and a BODS job starts in this  
duration, it will not be identified as a new job.  
5. Basic level testing has been done, however extensive testing is still  
required.  
6. Exception Handling has not been implemented everywhere.  
7. If user opens the data files while the program is running, error occurs.