3



Application Monitoring User Manual



Version 2.0

November 27, 2019

**Table of Contents**

[Document Version History 3](#_Toc14724438)

[1.0 Introduction 4](#_Toc14724439)

[2.0 Overview 4](#_Toc14724440)

[3.0 Related Documents 4](#_Toc14724441)

[4.0 Conventions 4](#_Toc14724442)

[5.0 Approach 5](#_Toc14724443)

[6.0 Getting Started 6](#_Toc14724444)

[6.1 Set-up Considerations 7](#_Toc14724445)

[6.2 Accessing the System/Application 7](#_Toc14724446)

[6.3 How to Configure 8](#_Toc14724447)

[6.4 Framework Extension 8](#_Toc14724448)

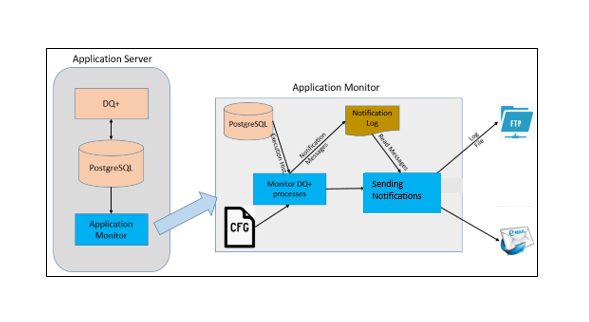
6.5 Assumptions……………………………………………………………………………………………8

[7.0 Troubleshooting & Support 8](#_Toc14724449)

# Document Version History

| Author(s) | Version | Amendments/Reasons | Date |
| --- | --- | --- | --- |
| Sujata Kumari | 1.0 | Application Monitoring User Manual | 26-August-2019 |
| Atul Sharma | 2.0 | Addition enhancement to stage delay notification and email body | 27- November 2019 |

# Introduction

This User Manual provides the information necessary for DQ+ Operation team to effectively use the Application Monitoring solution to monitor the DQ+ processes and send appropriate notifications

# Overview

Application monitoring component is a standalone Java application deployed on application server where DQ+ application and PostgreSQL database are hosted. It monitors DQ+ processes, generates log file from the observations and send email notifications.

Whenever any PM, analysis or data view is executed in DQ+, its execution history is maintained in PostgreSQL database. This application monitoring component will rely on execution history to send notifications. This component will monitor only the first level processes (be it scheduled or manually triggered process) and not the intermediate processes.

As part of this solution, following notification events have been identified. On the occurrence of any of the below events, notification will be sent

* Any Process Model, analysis or DV failed due to technical reasons
* Any analysis or DV is in waiting state for more than x minutes
* Any PM, analysis or DV didn't execute as per expected frequency
* Any PM, analysis or DV is taking more than average execution time.

# Related Documents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sl.No | Title | Author | Date | Issue |
| 1 | Application Monitoring and Notification v0.3.docx | Ajay Gera | 20/11/19 |  |
| 2 | PostgresQueries.docx | Shiv Mangal Rahi | 27/11/19 |  |

# Approach

This solution uses below two files:

app\_properties.xml : It stores all configuration related information needed for the component.

Eg:

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>

<entry key="env\_name">TESTING</entry>

<entry key="email\_host">ex2010.infogix.com</entry>

<entry key="ftp\_server">perfln6</entry>

<entry key="log\_ftp\_path">/opt/log-project/notificaton\_log\_</entry>

</properties>

app\_control\_file.txt : This file is used to capture the timestamp when the component was last executed.

This component performs the below steps:

1. Read the properties file/control file for configurational information/last executed timestamp
2. Connect to PostgreSQL and execute the queries
3. Capture the observations in a log file
4. Send email notifications to appropriate recipients
5. Transfer the log file from host server to some destination server (FTP)
6. Overwrite the control file with latest timestamp

Log file structure:

It is pipe delimited file with below details:

Env Name

Stage Name

Stage Type

Process Id

Work Id

Alert Type

Email body structure:

It contains below details:

Stage Name

Stage Type

Process Id

Work Id

Execution Start Timestamp

Execution End Timestamp

Stage URL

Average Execution Time

Minimum Execution Time

Maximum Execution Time

Below scenarios are developed as part of Application Monitoring.

**Any PM ,Analysis or DV failed during execution:**

This notification will be sent whenever any scheduled process model fails in execution and its status in execution history is marked as ‘Failed’. It will check for processes failed after the last execution of the component

**Any analysis or DV is in waiting state for more than x minutes:**

This notification will be sent when any analysis or DV’s execution job is submitted but is in waiting state. A process may initially go in waiting state if maximum parallel running processes threshold has reached and new process can’t be initiated. If process is in waiting state for more than x minutes, component will send an alert. This alert will be sent only if waiting time is more than x minutes and process is in waiting state when this component executes. ‘x’ minutes threshold will be maintained in configuration file.

**Any PM, analysis or DV didn't execute as per expected frequency:**

Notification is sent when a scheduled instance of a process gets missed and doesn’t execute. DQ+ attach a calendar with each scheduled process or stage.

**Any PM, Analysis or DV taking more than its normal execution time:**

Notification is sent when any scheduled Process Model takes more than average execution time in last X execution. If there is no prior execution for a process, this notification won’t be applicable for it.

# Getting Started

The standalone Java application contains the below source files:

* *TriggerAppMonitoring.java* : Triggers the application monitoring process
* *DbConnectionManager.java* : Connects to PostgreSQL and executes queries
* *SQLStatements.java* : Contains SQL queries to fetch the relevant records as per scenarios
* *BaseNotificationOb.java* : Notification object structure
* *PropertyFileReader.java* : Reads configuration file and control file
* *GenerateLog.java* : Write observations into log file, perform FTP, overwrite control file with latest timestamp
* *PopulateNotificationOb.java* : Populate Notification object and send email notifications
* *MmtJobExpandNotificationOb.java:* MMTJobscheduleExpanded notification structure.
* *Cal.java*: Interface which contains abstract method.
* *DailyRecurringEntry.java*: Notification object structure for Daily Recurring Entries from the xml stored in defcontentsegment.caf\_defcontentpart
* *MinuteRecurringEntry.java:* Notification object structure for Daily Recurring Entries from the xml stored in defcontentsegment.caf\_defcontentpart
* *log4j.properties*: Property file for Generating logs for all the events when the application is running.

External JAR files used in this component:

* apache-commons-net.jar
* commons-logging-1.1.3.jar
* commons-vfs2-2.0.jar
* hamcrest-core-1.3.jar
* jsch-0.1.54.jar
* mail.jar
* postgresql-42.2.5.jar
* log4j.jar

The above files are exported into a runnable JAR i.e. AppMonitoring.jar

Unix script executeAppMonitoring.sh executes the above jar file, which triggers the application monitoring component.

Export AppMonitoring.jar, app\_control\_file.txt, app\_properties.xml, executeAppMonitoring.sh into a zip file e.g. D3SUTL\_APPLICATION\_MONITORING\_IGX\_2.0.0.zip

### 6.1 Set-up Considerations

Server Setup:

1. Createan application home folder e.g. AppMonitoring
2. Create subfolders config, script and log
3. Download the D3SUTL\_APPLICATION\_MONITORING\_IGX\_3.0.0.zip file and extract it
4. Copy the AppMonitoring.jar file under /AppMonitoring
5. Copy the app\_properties.xml and app\_control\_file.txt under /AppMonitoring /config
6. Configure APP\_HOME in executeAppMonitoring.sh e.g. export APP\_HOME=/home/sagacity/AppMonitoring
7. Copy executeAppMonitoring.sh under the script folder /AppMonitoring /script

PostgreSQL Setup:

1. Creation of custom schema and user
2. Providing grants to the user to enable it to access (read only) sagacity tables
3. Creation of tables in custom schema to be used by monitoring tool

For more details, please refer PostgresQueries.docx

### 6.2 Accessing the System/Application

1. Please contact Infra team to get read/write access to the unix box and PostgreSQL database deployed on the application server

### 6.3 How to Configure

Below steps to be followed:

1. Createan application home folder e.g. AppMonitoring
2. Create subfolders config, script and log
3. Download the D3SUTL\_APPLICATION\_MONITORING\_IGX\_3.0.0.zip file and extract it
4. Copy the AppMonitoring.jar file under /AppMonitoring
5. Copy the app\_properties.xml and app\_control\_file.txt under /AppMonitoring /config
6. Configure APP\_HOME in executeAppMonitoring.sh e.g. export APP\_HOME=/home/sagacity/AppMonitoring
7. Copy executeAppMonitoring.sh under the script folder /AppMonitoring /script
8. Give permission to the script using the command chmod -R 777 executeAppMonitoring.sh
9. Execute the script using below command

./ executeAppMonitoring.sh

### 6.4 Framework Extension

This solution can be extended/modified as per custom notification requirements. It will require creation/modification of SQL query for the same and adding method to execute this query.

### 6.5 Assumptions

The below test cases were not executed as there is no option of manually switching the scheduler on and off.

* To verify that application monitoring script monitors and creates the notification log with the details of any PM/Analysis didn't execute as per expected frequency and then notifies via email/ftp.
* To verify that application monitoring script doesn’t sends notifications for already notified alert when re-executing the monitoring script again for PM/analysis/dv didn't execute as per expected frequency

The SMS Notification is not delivered due to the complexity.

# Troubleshooting & Support

In case of any exceptions during the execution of the component, exception details along with stack trace is sent to appropriate recipients.

Log4j has been implemented which generate the log along with info and error in a log file