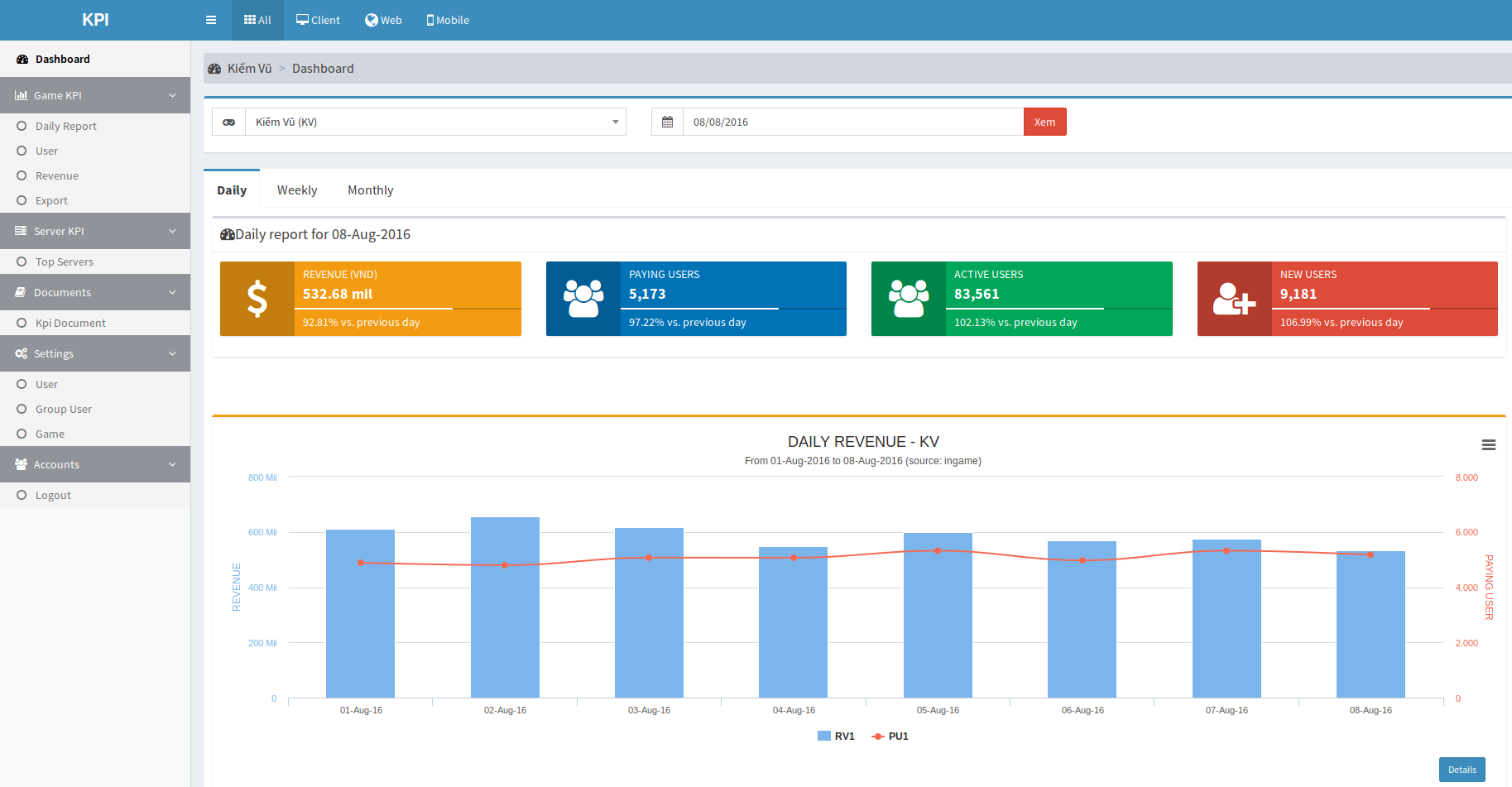
# Statistic’s Reporting System - Jobs Managment Documentation

Augst 09, 2016



<https://www.kpi.zing.vn>

STATS / FW & API Team

1. **Version history**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Update | Author |
| 2016-08-09 | 0.1 |  | vinhdp |
| 2017-02-27 | 1.0 |  | vinhdp |

1. **Overview**

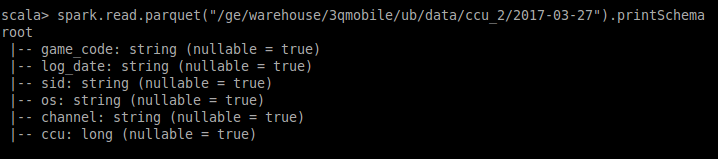
Statistic’s Reporting System is used to providing kpi report for game’s product and partners,...

URL: https://kpi.stats.vng.com.vn

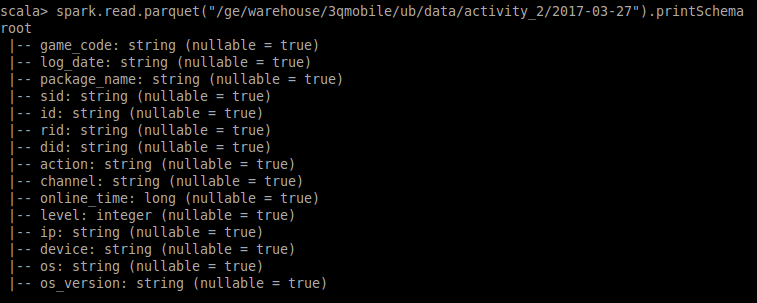
**Report type:**

|  |  |  |  |
| --- | --- | --- | --- |
| No | Name | GroupId | Kpi Name |
| 1 | Game | game | 1. CCU (only server kpi and game kpi) 2. Account Register 3. Active User 4. User Retention 5. New User Retention 6. New User + Revenue 7. Paying User + Revenue 8. First Charge + Revenue 9. First Charge Retention 10. Playingtime 11. Game Retention |
| 2 | Login Channel (group) | channel |
| 3 | Installed Package | package\_name |
| 4 | Server | sid |
| 5 | OS | os |
| 6 | Device | did |
| 7 | Country | country\_code |
| 8 | Hall Level (group mapping) | hall\_level |
|  |  |  |

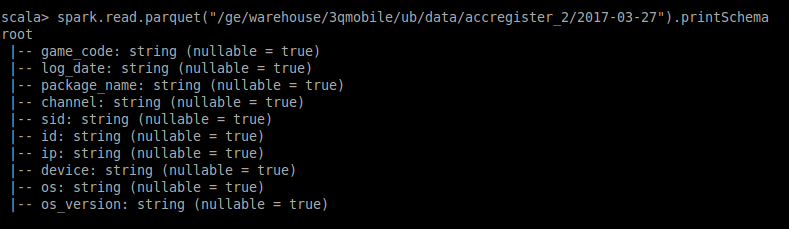
1. **HDFS Structure**
   1. **Raw log**
      1. **Structure**
         * /ge/gamelogs/[gameCode]/[logName]/[logDate]
   2. **Parquet log**
      1. **Structure v1**
         * /ge/warehouse/[gameCode]/ub/data/[logName]/[logDate]
         * /ge/warehouse/[gameCode]/ub/sdk\_data/[logName]/[logDate]
      2. **Structure v2 (fairy)**
         * /ge/fairy/warehouse/[gameCode]/ub/data/[logName]/[logDate]
         * /ge/fairy/warehouse/[gameCode]/ub/sdk\_data/[logName]/[logDate]
      3. **Schema**
         * **game\_code and log\_date** is **required for every** log
           + game\_code: game that occur the event
           + log\_date: the time that log was written (event occur time)
         * **sid is required for every log** **except** **total\_login\_acc\_2** and **total\_paid\_acc\_2**
           + sid: game has many server, each server has a unique id. Every log line was written from a server with server\_id (sid).
         * **ccu\_2**: store ccu log for a game.



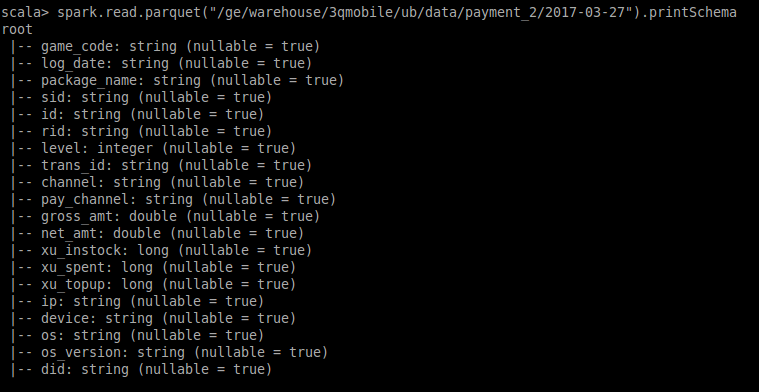
* + - * + os: operating system - describe as number of concurrent user that using this “os” online at the same time
        + channel: login channel - describe as number of concurrent user that using this “channel” online at the same time
        + ccu: number of concurrent user online at the same time (log\_date)
      * **activity\_2:** store log login and logout



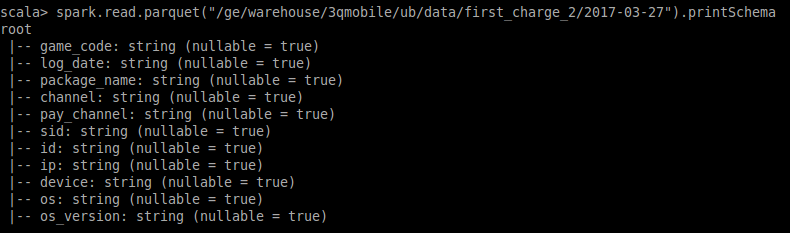
* + - * + package\_name: game has many version, each version from a package, each line log must be written from an unique package (id exist)
        + id: unique uesr on a game
        + rid: unique role id of a character on a server of a game
        + did: device id running game
        + action: action of user that create the activity: login/logout
        + channel: login channel: facebook, google, zalo, zingme,...
        + online\_time: total online time of an session, calculated from login until logout (online time usually written in logout action)
        + level: user level at the time log was written
        + ip: user ip
        + device: user device name (galaxy note 7, iphone 7s,..)
        + os: operating system (android, ios, bos,..)
        + os\_version: operating system version
      * **accregister\_2**: store log user register, user register is defined as the user that login into the game for the first time



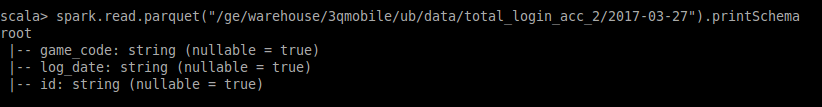
* + - * **payment\_2**: store paying log



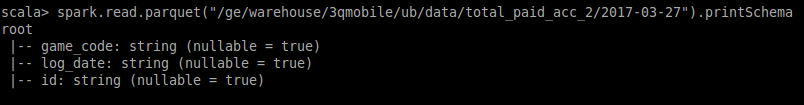
* + - * + trans\_id: every charging event has a unique transaction\_id for tracking
        + pay\_channel: user charging using an channel: paying channel: sms, amt, telco, zalopay...
        + gross\_amt: amount of money user charge into game
        + net\_amt: amount of money user recieve in game
        + xu\_instock:
        + xu\_spent:
        + xu\_topup:
      * **first\_charge\_2**: store log firstcharge, firstcharge is defined as user that paying for the first time in a game



* + - * **total\_login\_acc\_2**: store all unique user played game until log\_date

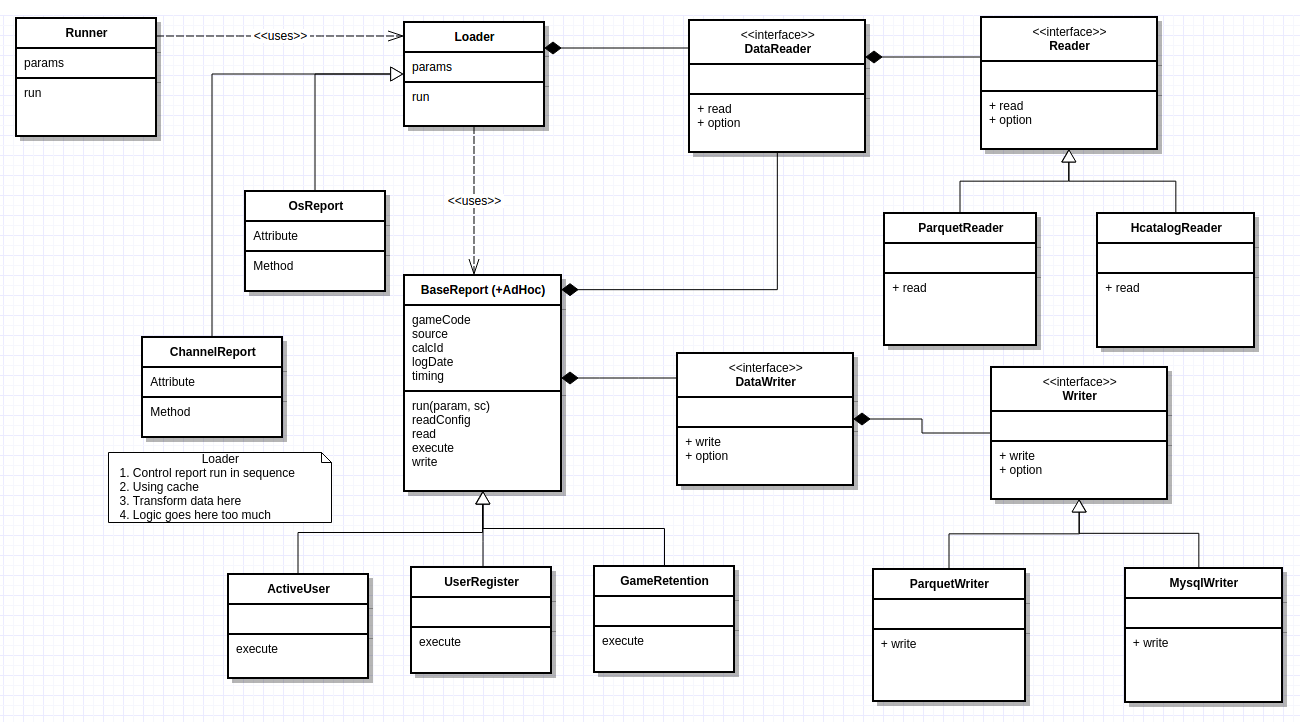


* + - * total\_paid\_acc\_2: store all unique paid user until log\_date



* 1. **Report**
     1. **Structure (MySQL):**

1. **Implementation**
   1. **Reporting**
      1. **Design**



* + 1. **Base**
       - **Package:**
         * vng.ge.stats.report.base.DataPool

Abstract class define common parameter using for loading file from HDFS, define how to load data (include filtering and transform data here) and which report will be calculated in order.

Implement new loader if we need new report or adding new report inside an loader body.

* + - * + vng.ge.stats.report.base.DataReader

Utility for file loader

Wrapper class from DataFrameReader

Added some function define how to load data from provided parameter (load one file, multi file by timing at a date)

* + - * + vng.ge.stats.report.base.DataWriter

Utility for writing report data to orther data source

Support writing dataframe, list of data (need to implicited convert element) to parquet, csv, mysql. Based on provided format (default is JDBC), choose an approriate instance for writing report using options provided.

* + - * + vng.ge.stats.report.base.TReport

Define common parameter using dirung report process

Define templete to run report

Implementation classes may reuse in many loader classed

* + 1. **Game Report**
       - **Package:** vng.stats.ub.report.report.game
       - **Class:**
         * AccountRegisterReport.scala
         * ActiveUserReport.scala
         * CcuReport.scala
         * FirstChargeReport.scala
         * FirstChargeRetention.scala
         * GameRetentionReport.scala
         * NewUserRetentionReport.scala
         * NewUserRevenueReport.scala
         * PaymentReport.scala
         * PlayingtimeReport.scala
         * UserRetentionReport.scala
       - **TBD**
    2. **Group Report**
       - **Using for calculating:** Login Channel, Installed Package, Server, Os, Mapping Report
       - **Package:** vng.stats.ub.report.report.group
       - **Class:**
         * GroupAccountRegisterReport.scala
         * GroupActiveUserReport.scala
         * GroupCcuReport.scala (for server only)
         * GroupFirstChargeReport.scala
         * GroupFirstChargeRetention.scala
         * GroupNewUserRetentionReport.scala
         * GroupNewUserRevenueReport.scala
         * GroupPaymentReport.scala
         * GroupUserRetentionReport.scala
       - **TBD**
    3. **tbd**

**TBD**

1. **Oozie jobs management**
   1. **Basic info**

* Local oozie script location:
  + Spark1: [stat-spark]/vinhdp/\*
    - Available at svn repos
  + Spark2: [stats-etlr]/vinhdp/\*
    - Available at git repos
* HDFS oozie script location:
* ***Host:*** 10.60.43.15
* ***User:*** fairy
* ***Location:*** /user/fairy/oozie/vinhdp/bundle/\*
* Oozie job submition location:
* ***Location:*** /home/fairy/ub/bundle/vinhdp/\*
* ***Info:*** info.sh
  + Show job info
  + Use: sh info.sh oozie’s id
* ***Kill:*** kill.sh (zkill.sh using for zdeploy user)
  + Kill running job
  + Use: sh kill.sh oozie’s id
* ***Submit:*** submit.sh (zsubmit.sh using for zdeploy user)
  + Submit new bundle job
  + Use: sh submit.sh [group] [type]
  + [group]: etl or report
  + [type]: report type
  1. **TBD**

1. **Build**
   1. **SBT assembly**

Using to builde all source code into on runnable jar.

* + 1. **Stats-spark.sbt build file**
       - **assemblyJarName:**

Output jar file’s name after assembly, the jar file is located on: [project\_source]/target/scala-2.11

* + - * **scalaVersion:**

Scala build version

* + - * **unmanagedBase:**

Using to include library when building with sbt, all lib in this location will be include in the final jar. **Current file using “unlib” is a cheat for eclipse error tracking**, default is “lib”.

* + - * **libraryDependencies:**

All library need in build process, get lib in maven repository tab SBT.

Ex: "org.apache.spark" % "spark-sql\_2.11" % "1.5.2" % "provided"

=> “groupId” % “artifactId” % “version” % “isProvided”

* artifactId must approriate with scalaVersion.
* isProvided = provided => this jar do not include in the final jar, if provided is removed, the jar will be include in the final jar.
  + - * **excludedJarsName:**

Jar file need to be excluded frm the final jar, unless it will be included in final jar

* + - * **excludedJars:**

Exclude function.

* 1. **Ant task**
     1. **Location:** [project root]
     2. **Build runnale jar**
        + Ant sbt-assembly: spark 1 project
        + Ant mvn-package: spark 2 project
     3. **Copy**
        + Copy jar and bundle script to HDFS
        + Params:
        + **group:**  etl or report
        + **type:** report type
     4. **Recopy**
        + Rebuild, copy jar and bundle script to HDFS
        + Params:
        + **group:** etl or report
        + **type:** report type
     5. **Example**
        + Copy etl sdk\_daily to HDFS:

**ant** copy -Dgroup=etl -Dtype=sdk\_daily

* + - * Build and copy report hall\_level to HDFS:

**ant** recopy -Dgroup=report -Dtype=hall\_level

* + - * Or build jar only:

**ant** sbt-assembly: for spark 1 in version 1

**ant** mvn-package: for spark 2 in version 2

1. **Notes**
   * 1. **Understanding report parameters and default values**

|  |  |  |  |
| --- | --- | --- | --- |
| **Param Name** | **Description** | **Default** | **Required** |
| game\_code | Game code in HDFS |  | yes |
| log\_date | Report date |  | yes |
| run\_timing | Used from job.Runner, control how report run over timings, each timing will be passed to loader to overwrite “timing” param at report. | a1,a3,a7,a14,a30,a60,ac7,ac30 |  |
| timing | Default at report, if report is running directly from loader without timing param (call function report directly) | a1 |  |
| calc\_id | Using this id for counting | id |  |
| source | Source file from HDFS: ingame, sdk, payment | ingame |  |
| group\_id | 1. “” or “game” for game kpi 2. “sid” for server kpi 3. “os” for os kpi 4. “channel” for channel kpi 5. “package\_name” for package kpi 6. Other => group mapping kpi | “” (empty string) |  |
| report\_number | Choose which report need to run | 1-2-3-4-5-6-7-8-9-12 |  |
| log\_dir | Root dir in HDFS using for create file path | /ge/fairy/warehouse |  |
|  |  |  |  |
| ccu |  | ccu\_2 |  |
| activity |  | activity\_2 |  |
| acc\_register |  | accregister\_2 |  |
| payment |  | payment\_2 |  |
| first\_charge |  | first\_charge\_2 |  |
| total\_login\_acc |  | total\_login\_acc\_2 |  |
| total\_paid\_acc |  | total\_paid\_acc\_2 |  |
| group\_mapping\_file | Define mapping file using in join action when getting group of an id |  |  |
| mapping\_field | Using when mapping file do not have required schema, mapping field will be renamed as group\_id |  |  |
|  |  |  |  |

* + 1. **How to create new kpi id**
       - Class **vng.ge.stats.report.util.IdConfig**
       - Each kpi (called “key”) mapping with a number: xxx00
       - Each timing mapping with a number (called “timing”): xx
         * A1 -> 1
         * A2 -> 2
         * A3 -> 3
         * A7 -> 7
         * A14 -> 14
         * A30 -> 30
         * A60 -> 60
         * AC7 -> 17
         * AC30 -> 31
       - KPI id for specific timing is calculated = “value of key” + “timing” in getKpiId(calcId, key, timing)
       - To create new kpi, add new key and mapping with a number that is not existed in the original list with required format xxx00 (simply increase xxx by 1 from the lastest)
       - Two more step to show kpi when export:
         * Add kpi id to kpi\_desc table in mysql
         * Add config key to db\_field\_config method on Util.php (ub-frontendvsmt project)
         * **kpi/daily & group kpi UI must add kpi\_id to source code to available**
         * Db\_field\_config info:

4 => A1

5 => A7

6 => A30

8 => A60

9 => A90

3 => A3

14 => A14

17 => AC7

31 => AC30

* + 1. **How report metric stored in database mysql**

|  |  |  |
| --- | --- | --- |
| **Report** | **Table** | **Note** |
| Game Kpi | game\_kpi |  |
| Game Kpi By Did | device\_id\_kpi |  |
| Server Kpi | server\_kpi\_json | Value is json: {sid -> value} |
| Os Kpi | os\_kpi |  |
| Channel Kpi | channel\_kpi/group\_kpi\_json |  |
| Package Kpi | package\_kpi/group\_kpi\_json |  |
| Country Kpi | country\_kpi\_json | Value is json: {country -> value} |
| Hall level + mapping Kpi | group\_kpi\_json | Value is json: {group -> value}  group\_id: Using for tracking report type |
| Game Retention | game\_retention |  |
| AppsFlyer | marketing\_kpi |  |

* + 1. **How to implement new report**
       - Write new Runner or expand by adding new group id
       - Write new Loader by extending DataPool
       - Write new Report by extending TReport
       - Write new Writer by extending Writer class or expand DataWriter
       - **FLOW: (top-down)**
         * Runner: define how to deal with run timing, log date, report number,.. Or everything we define (only run on weekend, end of month,...)
         * Loader: Resposibility for loading file from HDFS. May be using for join, filter, transform, caching before calling other report
         * Report: Excute main report logic. May be reuse many time
         * Write: Write report results to external storage like HDFS, MySql
    2. **How to setup new group mapping report**
       - Sample: hall\_level
       - Using ETL create new file in HDFS with schema (game\_code, log\_date, id, group) in default folder (ub/data).
       - Setup as hall\_level, modify below params:
         * **group\_id**: id we want to use in url /group/{group\_id}
         * GroupMappingDataLoader automatically convert “group” field in mapping file to “{group\_id}” field after join
         * In bundle.xml modify parameter mappingFile to our mapping file in HDFS
         * In case of our file is not have schema as default schema, we need add new param for mapping loader select right field. Add param “mapping\_field” to new field in mapping file (like sdk\_country\_kpi)
       - **Adding report title and prefix before group\_id in UI:**
         * insert into group\_name\_conf values (“group”, “{group\_id}”, “{json\_config\_value}”, NULL, “2017-03-22”, 1)
         * **group\_id**: id using to identify report.
         * **json\_config\_value**: {"name": "{report\_title}", "prefix": "{prefix\_before\_group\_id}"}
    3. **How to rename group in group report UI**
       - Using to define **alias** for an **group id** in database when showing data on **group report, channel, package, server** report UI
       - Config mapping id in table **group\_name\_config**:
         * **Game\_code:** configuration apply for **this “game\_code”.** If **game\_code = “allgames”**, this configuration will be applied for all games
         * **Id: group\_id** in database
         * **Name: alias** using to display in UI
         * Merge\_id: not important when defining alias, may be NULL
         * Date: create date
         * Active: is active (currently not used)
       - Ex1: If we want rename group\_id = “android” to sentences case in UI for all games:
         * insert into group\_name\_config values (“allgames”, “android”, “Android”, NULL, “2017-03-22”, 1)
       - Ex2: If we want “S52” showing on UI instead of id = “13052” in server report for only game “dttk”:
         * insert into group\_name\_config values (“dttk”, “13052”, “S52”, NULL, “2017-03-22”, 1)
    4. **How to config merge server**
       - Using to map old “server\_id” into new “server\_id” before calculating group report. **Implementation in ServerDataLoader**
       - **This configuration must be inserted into database before server report run.**
       - Config mapping id in table **group\_name\_config**:
         * **Game\_code: configuration apply for this “game\_code”**
         * **Id: old server\_id**
         * Name: name using to display in UI
         * **Merge\_id: new server\_id**
         * Date: create date
         * Active: is active (currently not used)
       - Ex: If we want merge server 13051 and 13052 into server 13052 for game dttk:
         * insert into group\_name\_conf values (“dttk”, 13051, “S51”, “13052”, “2017-03-22”, 1)
         * insert into group\_name\_conf values (“dttk”, 13052, “S52”, “13052”, “2017-03-22”, 1) => do not required because 13052 map to 13052 - nothing change
    5. **How to fix missing ccu report for game DTTK**
       - DTTK ccu log using server name instead of server id => mapping server name back to server id in UI
       - Using game\_code = “dttk\_fix”, insert mapping as rename group
       - Ex: if we want mapping server with name Shinto to id 13053 with name S53:
         * **Map name to id:** insert into group\_name\_conf values (“dttk\_fix”, “Shinto”, “13053”, NULL, “2017-03-22”, 1)
         * **Map id to new name:** insert into group\_name\_conf values (“dttk\_fix”, “13053”, “S53”, NULL, “2017-03-22”, 1)
    6. **How to add new report to UI**
    7. **How to config global payment**
       - Using to:
         * Map application code to UB’s game code
         * Map item\_id (payment) to UB’s game\_code
       - Using in **vng.stats.ub.normalizer.GlobalFormatter**
       - Config **file location**:
         * Local file config: **stat-spark/conf/**
         * Hdfs file config: **/ge/gamelogs/global/**
         * dbgend\_app.csv:

Using to mapping appId from dirrect billing to ubCode (game\_code in report system)

Schema:

|  |  |  |
| --- | --- | --- |
| Field | Required | Desc |
| appId | yes | Direct Billing App ID |
| appName |  | App name |
| ubCode | yes | Game code using in report system |
| AppDesc |  | App desc |
| IsMobi |  | Is it a mobile game? |

* + - * + **zingxu\_item.csv**:

Using to map **item** (zingxu item) to **pid** (product id) in convert/converdetail file

Schema:

|  |  |  |
| --- | --- | --- |
| Field | Required | Desc |
| ProductID | yes | Zingxu product id |
| ProductCode |  | Zingxu product code |
| ProductName |  | Zingxu product name |
| ItemId | yes | Zingxu item id |

* + - * + **zingxu\_product.csv**:

Using to map **pid** (product id) to UB **game\_code**

Schema**:**

|  |  |  |
| --- | --- | --- |
| **Field** | **Required** | **Desc** |
| ProductID | yes | Zingxu product id |
| Payment |  | Payment code |
| UB\_CODE | yes | UB game\_code |
| ProductName |  | zing xu product name |

* + 1. **How to convert old chanel table to new group table**
       - Convert data from table channel\_kpi to group\_kpi\_json (like nikkisea: group/channel)
       - Controller: **GroupDataConvert (stats-ui)**
       - Select game need to convert
       - Call java script function: convertData(“group\_id”, “fromDate”, “toDate”)
         * group\_id: channel
         * fromDate: YYYY-MM-DD
         * toDate: YYYY-MM-DD
    2. **Operation task**
       - Check log
         * Check if report is enough, duplicate payment, miss server log...:

See problem game at <https://kpi.stats.vng.com.vn/index.php/operation/overview> and cross check mail, daily report mail,..

Order by log\_date, check if a user whose paying info is not different at a same time

Group log by server\_id and compare to previouse day to see the difference

* + - * + Recieve alert from product owner
      * Rerun:
        + Using **spark-submit**: configuration the same as using oozie (above configuration), can be using bash script looping day by day. Appropriate for rerun a **large range date**

Some **spark-submit config** can be found at: **stats-etlr/scripts/spark-submit.sh**

* + - * + Using **oozie** rerun for **small date range** (about 2 date):

Run for a game (run **coordinator**):

Must be run **in order** if job rerun is **ETL task**

**oozie job -rerun** *[coordinator\_id]* **-action** *[number]*

Run for all game (run **bundle**):

Can not use for ETL

Oozie job -rerun *[bundle\_id]* -date *[fromDate]***::***[toDate]*

*fromDate, toDate format: YYYY-MM-DD****T****HH:mmZ*

* + 1. **TBD**

1. **TBD**