**设备应用安装列表开发手册**

## 一、数据字典

#### 1.1表级

英文名：dwd\_onl\_device\_app\_install\_statis\_dm

中文名：设备应用安装统计表

表类型：全量表

ETL调度频率：日表

#### 1.2字段级

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 字段中文名 | 备注 |
| imei | VARCHAR(128) | 设备编号 |  |
| package\_name | VARCHAR(256) | 包名 |  |
| app\_ver | VARCHAR(128) | 应用版本 |  |
| first\_install\_time | VARCHAR(30) | 首次安装时间 |  |
| last\_update\_time | VARCHAR(30) | 最近更新时间 |  |
| first\_report\_time | VARCHAR(30) | 首次上报时间 |  |
| uninstall\_flg | SMALLINT | 卸载标志 1卸载 |  |
| data\_src\_cd | SMALLINT | 数据来源代码（应用市场1,大数据采集2,两个都有3） |  |
| etl\_time | VARCHAR(30) | 设备编号 |  |

## 二、数据来源

#### 2.1 2个来源:

dwd\_evt\_user\_install\_list\_dm (应用市场)

dwd\_evt\_bdreporter\_app\_info\_report\_dm (大数据采集)

## 三、数据逻辑

#### 3.1主体逻辑

#@T1 应用市场

#@T2 131073 大数据采集安装列表

#@T3 131074,131076状态

#@T4 131075状态为卸载， MAX(record\_time )作为uninstall\_time

#@T5 取昨日数据 uninstall\_flg为1则把last\_update\_time赋值给卸载时间uninstall\_time

#@TT2 今日大数据采集上报的imei号及对应的first\_report\_time

#@TT3 取昨日的first\_report\_time

T1到T5做union all处理，app\_ver和last\_update\_time都做了打横处理为今日和昨日两个字段，并每段加上t1\_flg到t5\_flg标志字段。

#### 3.2字段逻辑

| 字段名 | 处理逻辑 |
| --- | --- |
| imei | 主键。格式规范化，小写。 |
| package\_name | 主键。由于同一个安装包包名同时存在大小写，所以统一处理为小写。 |
| app\_ver | 当前最新状态对应的版本号。 |
| first\_install\_time | 首次安装时间，第一次统计到这个package安装的时间，比如之前安装了后来卸载了，今天又安装，first\_install\_time为首次安装的时间，不会更新成今天。 |
| last\_update\_time | 最近更新时间  #今天没有上报数据，更新时间为昨天；  #今天版本为空,没有操作行为,有昨天和今天的列表,更新时间不变；  #今天有上报，版本无变化且昨日今日都未卸载,更新时间不变；  #最后更新时间为空即昨天有应用,今天安装列表没有上报应用,即为已卸载,标记今天时间；  #其他场景均为今天更新时间。 |
| first\_report\_time | 首次上报时间，是对应于imei的首次上报时间，  即同一imei，该imei以及对应package的first\_report\_time都一样 |
| uninstall\_flg | #4.1.1 卸载时间不为空，卸载时间大于更新时间为卸载；  #4.1.2 今天没有上报列表，昨天是卸载的保留卸载标识；  #4.1.3 今天有上报列表，昨天的应用没有上报(更新时间为空)为卸载； |
| data\_src\_cd | 1是只来源于应用市场，2是只来源于大数据采集，3是表示来源于两个表  详见4.3 |
| etl\_time | ETL时间。 |

## 四、处理逻辑及常见问题

#### 4.1 卸载标志uninstall\_flg

4.1.1 卸载时间不为空，卸载时间大于更新时间为卸载；

即取record\_time最大时间对应的状态作为最终的状态，比如先安装（更新）后卸载；

4.1.2 今天没有上报列表，昨天是卸载的保留卸载标识；

比如昨天有这个imei，今天没有上报这个imei，我们认为是漏报，uninstall\_flg直接取昨天状态；

4.1.3 今天有上报列表，昨天的应用没有上报(更新时间为空)为卸载；

比如昨天上报了imei+package，但今天上报了imei，没上报该package，（例如昨天这个imei上报了100个package，今天上报了99个），我们认为这个package是卸载了。

（正是由于有这个逻辑，各个应用在统计当前安装量的时候，c30没有对这种场景的判断和处理，认为还是安装，而我们处理是卸载）

（当前安装量**筛选逻辑**：where package\_name=’XXXX’ and uninstall\_flg=0）

同时，统计新增安装量的时候，以前安装过后来卸载了，今天再次安装的，c30认为是新增安装，而我们认为是历史安装，不属于新增(first\_install\_time不是当天)**。**

（新增安装量**筛选逻辑**：where package\_name=’XXXX’ and uninstall\_flg=0 and TO\_DATE(first\_install\_time)= '$date\_ep'）

#### 4.2 record\_time处理

安装列表和大数据采集都有脏数据，统计时间字段大于当前日期，且数据量不小，我们没有删除，是处理为当天日期+ report\_time(rec\_time)的时分秒

IF(TO\_DATE(report\_time)<='$date\_ep',report\_time,CONCAT('$date\_ep',SUBSTR(report\_time,11,9))) AS report\_time

IF(TO\_DATE(rec\_time)<='$date\_ep',rec\_time,CONCAT('$date\_ep',SUBSTR(rec\_time,11,9))) AS record\_time

由于最终的卸载标志是用各个状态的record\_time做比较，取最大时间对应的状态，所以这些脏数据会导致卸载标志有误差

***综合4.1、4.2所述，卸载标志的处理逻辑和c30有很大差异，c30是有卸载行为才算卸载，没有对以上各种场景的处理，也没有对report\_time脏数据的处理逻辑，所以请以c60为准。***

#### 4.3来源data\_src\_cd处理逻辑

1是只来源于应用市场，2是只来源于大数据采集，3是表示来源于两个表，

如果只要统计大数据采集的，那就要筛选来源**（筛选逻辑：where data\_src\_cd in (‘2’,’3’)）**

MAX(data\_src\_cd)|MIN(data\_src\_cd) AS data\_src\_cd

按位或是位运算的一种，是将两个数据的二进制表示右对齐后，按位进行运算，两个对应的二进制位中只要一个是1，结果对应位就是1。当前是2个来源，多个来源也可以满足，但需要打横后处理。

三个来源分别是1,2,4

…

1|2=001|010=011=3

1|4=001|100=101=5

2|4=010|100=110=6

1|2|4=001|010|100=111=7

…