元数据应用开发规范指导

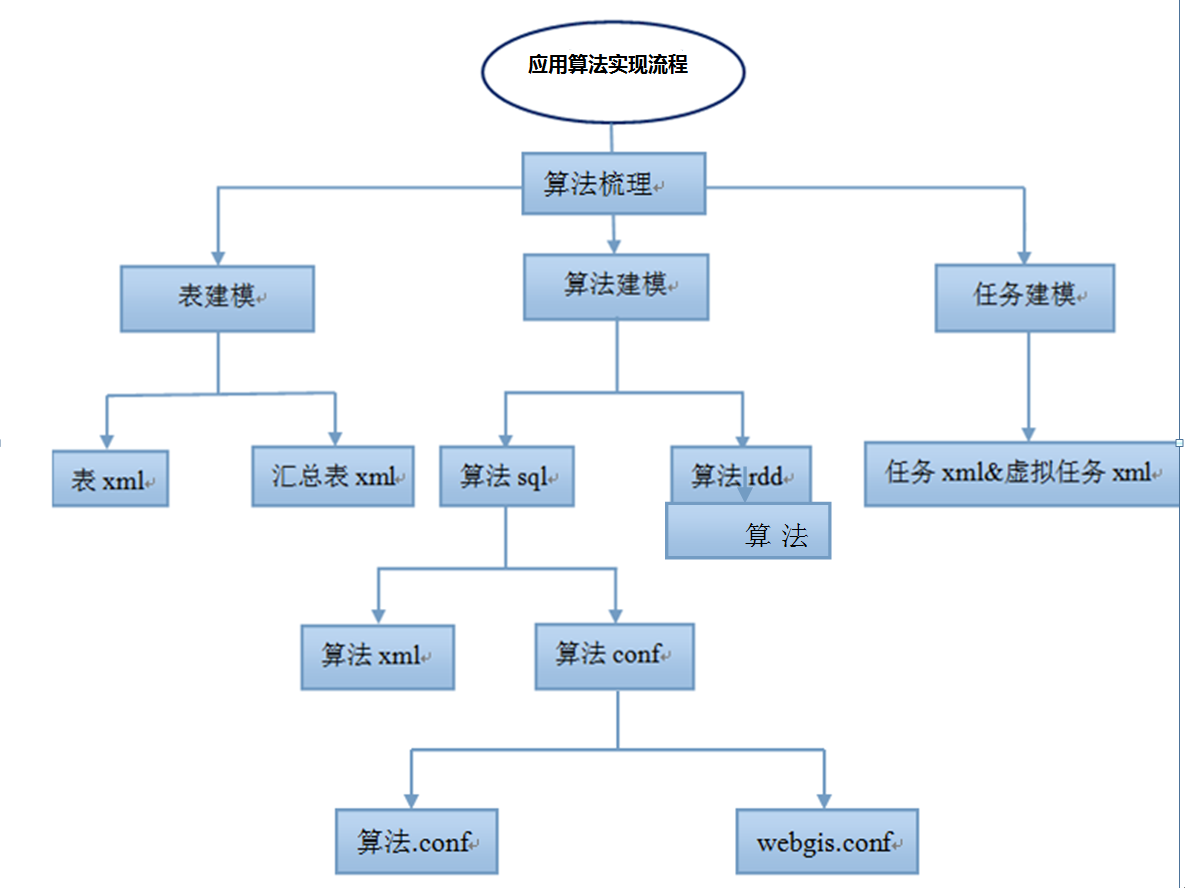
**修改记录**

| 版本号 | 拟制人/\  修改人 | 拟制/修改日期 | 更改理由 | 主要更改内容  （写要点即可） |
| --- | --- | --- | --- | --- |
| v0.1 | 李鹤 李勇 赵银玲 | 2015/12/16 | 新建 | 无 |
| V1.0 | 王会 | 2016/04/06 | 修改 | 基于平台2016.4.18日发布的ZTE VMAX Metadata Manager V6.15.50.03-0418版本整理 |
|  |  | 2016/04/26 | 修改 | 增加sql文件不能使用UTF-8（无BOM编码格式）的说明 |
|  |  | 2016/05/18 | 修改 | 1. 5.7增加数据公告牌可视化分类配置 2. 5.8增加不可到达netmax表配置 |
|  |  | 2016/05/30  （Sprint51） | 修改 | 算法xml中subtype字段增加支持gbase数据驱动任务的类型：gbaseload、gbasejar、gbasesql |
|  |  | 2016/06/3  （Sprint51） | 修改 | 章节6表升级规范说明中，安装系统管理的升级规范要求进行说明，主要去掉支持字段类型升级等 |
|  |  | 2016/06/21  （Sprint53） | 修改 | 1. 修改RDDjar包说明 2. 修改数据生命周期配置说明 |
|  |  | 2016/06/21  （Sprint54 | 修改 | 1）3.1.6章节汇总表xml说明classify字段取值范围增加fixednetwork, serviceexperience, iptv, multinetwork, health  2）3.1.4章节任务xml说明中修改type字段取值范围，增加userdriver，用于支持用户驱动任务功能  3）增加5.9 栅格大小配置说明  4）章节5.3数据到达检查相关配置增加iptv和broadband两种类型的检查规则说明，同时修改了汇总表xml中datasourcetype的取值范围 |
|  |  | 2016/07/28  （Sprint56） | 修改 | 1)3.1.1章节算法XML文件说明中修改spark\_special\_parameter属性的配置说明，规范配置参数的文件名格式，增加特殊参数的配置说明  2）增加5.10章节推送表文件到指定ftp服务器的配置说明  3）修改章节3.1.6汇总表xml说明和5.3数据到达检查配置说明，增加datasourcetype为streaming\_partition的说明 |
|  |  | 2016/08/24  （Sprint57） | 修改 | 1）插入章节4 spark streaming流任务的配置说明  2）修改章节3.3，增加公共sql提供latestDayOfPartition获取接口  3）章节3.1.5表xml文件说明和3.1.6汇总表xml说明字段tabletype增加gbase\_dynamic类型，支持核心网分表  4）章节6.3数据到达检查配置增加streaming\_partition和spark\_txt\_file2检查方式 |
|  |  | 2016/09/07（Sprint58） | 修改 | 1）章节3.1.1.1 算法xml文件说明字段说明，字段executecycle增加checkpointrecovery标签，作用： 是否在上电时生成该时间点以前的当天所有定时任务。（gbase/spark准实时定时驱动任务） |

# 文档说明

本文档用于对基于元数据平台的应用开发使用指导，用于说明应用特性如何在元数据平台上进行功能开发和应用实现的注意事项。

# 应用算法实现概述



应用算法实现主要包括表建模，算法建模和任务建模三部分。这三部分的工作可以同步进行。其中，

1. 表建模主要是完成表的xml文件和汇总表的xml文件，表xml文件对应生成表的建表脚本，汇总表xml说明表的建表路径等信息；
2. 算法建模主要分两种形式：sql和rdd。

* 对于通过sql实现的算法，需要配置算法.sql文件，算法.xml文件和算法.conf文件，这三种文件相互关联组成一个实例化的sql算法
* 对于通过rdd实现的算法，需要打成jar包；

1. 任务建模包括任务xml文件和虚拟任务xml文件的生成
2. 应用算法实现步骤说明如下：

| **工作步骤** | **工作** | **关注点** | |
| --- | --- | --- | --- |
| **业务梳理** | etl算法流程梳理 | 注意事项：  1）算法粒度尽量小：  一个算法就做一件事情；  算法事件涉及的运行环境要单一：只有spark或sybase或hbase等，如从spark清洗后又同步到sybase，则拆开，目前只实现spark-spark即可，同步后续再统一考虑；  2）系统目前只支持spark->spark：因此算法中有如sybase-sybase这种，需要修改为由spark清洗，清洗后再同步到sybase  3）算法形式为.sql或jar包：若纯sql则以.sql形式提供，若使用了rdd则以jar包形式提供，若使用了sql和rdd混合形式则以jar包形式提供  4）临时表处理：  a）若临时表只是给本算法使用，则该临时表放到算法内，同放在一个.sql文件中，或同放到一个jar包中，且算法xml中输出表名没有该临时表   b）若一个job中产生的临时表被多个job共用，则该临时表需要独立成正式表; | |
| 输入：  1）方案文档（如果没有请脑补）；  2）功能代码（UT输出的SQL以及代码），特指一个功能（比如cdma 1x综合分析） 输出：  1）给出需要被建模的表的列表（含NetMAX主题表、清洗中间生成的表、SybaseIQ的表、Hbase暂时不必建模）  2）给出需要被建模的算法列表：将当前的功能代码中SQL分拆为多个算法。  3）文本文件：包含功能相关的所有UT生成的SQL代码； | |
| **表建模** | 生成表的xml文件 | 对应设计的表填写表xml文件；详细说明见章节[表XML文件说明](#_表XML文件说明) | |
| 生成汇总表的xml文件 | 对应表存放的路径，数据库等类型填写汇总表xml文件，详细说明见章节[汇总表xml说明](#_汇总表xml说明) | |
| **算法与检查算法建模** | 生成算法.sql文件 | 算法建模主要分两种形式：sql和rdd，在算法xml配置算法类型subtype用于区分。  对于通过sql实现的算法，需要配置算法.sql文件，算法.xml文件（元数据说明文件）和算法.conf（一些需要配置的参数）文件，这三种文件相互关联组成一个实例化的sql算法；对于通过rdd实现的算法，需要打成jar包；配置算法.xml文件和算法.conf文件等，由算法jar包自行完成算法的实例化。  算法拆分拆分的原则：算法尽量小，且一个算法内的sql执行体，必须同属于一种执行环境，如都基于spark运行 | 根据算法逻辑实现算法sql文件，将算法的sql语句摘取出来，支持一个.sql文件内有多个sql执行语句，以分号分隔；详细说明见章  节[Sql文件说明](#_Sql文件说明) | |
| 生成算法  .rdd的jar包 | 对于使用sql文件无法实现的算法逻辑或需要调用特殊jar包的，使用RDD完成的算法，需要将代码生成jar包，系统通过执行jar包  完成算法，详细说明见章节[Rdd jar包说明](#_Rdd_jar包说明) | | |
| 生成算法.xml文件 | 输入：整理算法运行依赖的表和分区 输出：算法输出的表，可以有多张 *备注：1）算法中输出表和依赖的输入表相同时，不作为入参，如周利用昨天周+今天天，周表不作为入参，同时，此表使用的分区，需要依赖其他表的分区计算 2）对于需要同步到sybase、GBase、Hbase等数据库的表，需要配置相应的\*\_sybase.xml、\*\_Gbase.xml或\*\_Hbase.xml文件，算法格式基本一致，通过subtype算法类型实现区分。3）对于需生成webgis预处理文件的算法，需配置\*\_webgis.xml和\*\_webgis.conf。*  详细说明见章节[算法XML文件说明](#_算法XML文件说明) | | |
| 生成算法.conf文件 | 算法实现中需要使用的配置参数等存放的文件；原定每个算法一个.conf文件，业务可根据实际需要灵活确定，比如一个功能一个也可。  详细说明见章节[Conf文件说明](#_Conf文件说明) | | |
| **任务建模** | 生成任务xml & 虚拟任务xml文件 | 应用功能实现时的多个算法形成一个任务或虚拟任务，形成任务xml，详细说明见章节[任务XML文件说明](#_任务XML文件说明) | | |

# 开发实现文件说明

## XML文件说明

元数据平台下有多种XML文件，用于描述算法、任务、表等的元数据属性。基于元数据平台的开发过程，就是XML文件的建模过程，必须正确配置各类XMl文件才能实现算法功能。以下将对各类xml的内容字段的用法进行说明，并配以示例说明。

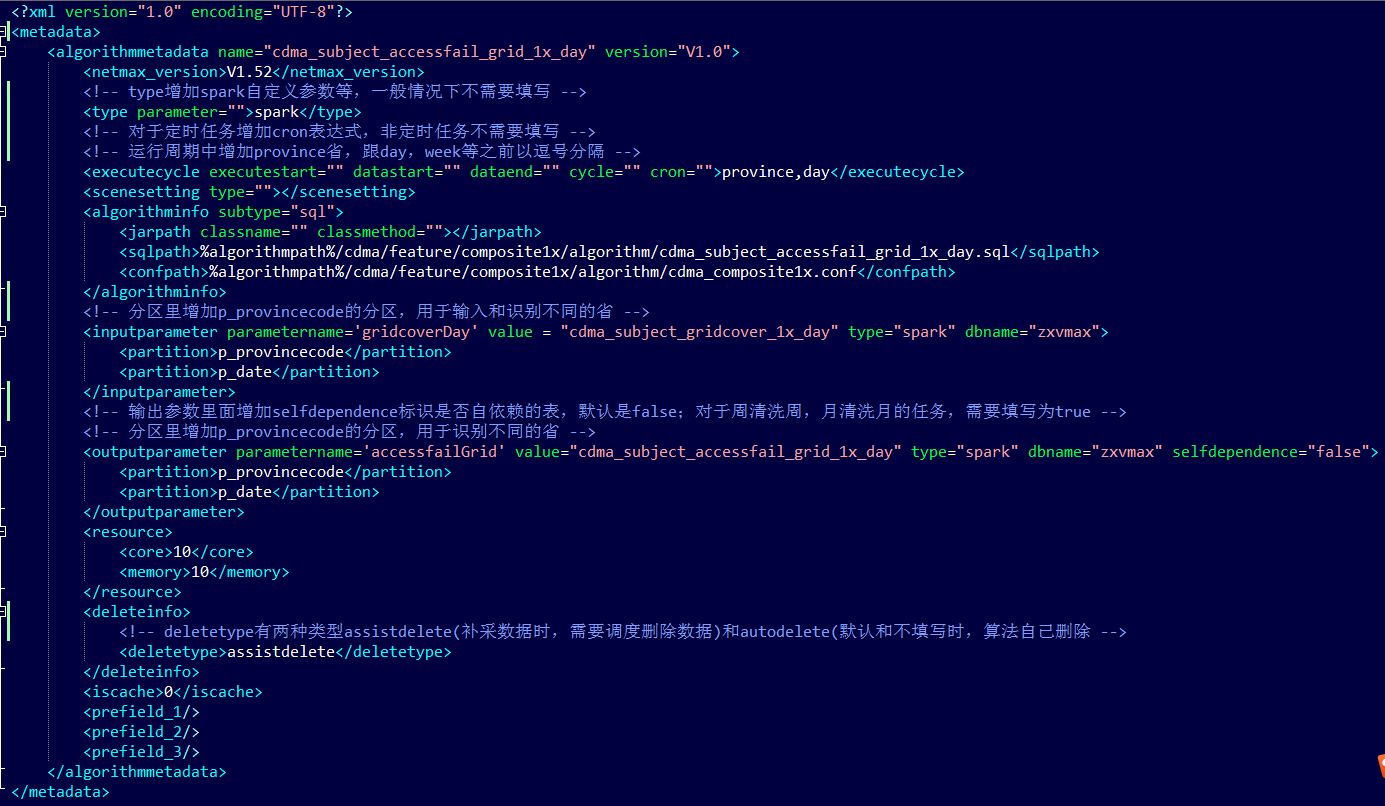
### **算法XML文件说明**

#### 字段说明

| **字段名** | **子字段** | **字段说明** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| name |  | 算法名 | string |  | umts\_overcoverage | 与生成表名一致 | 6.15.50.03-0418 |
| version |  | 算法版本号 | string |  | V1.0 | 算法版本号 | 6.15.50.03-0418 |
| netmax\_version |  | 主题表版本号 | string |  | V1.52 | netmax主题表版本号 | 6.15.50.03-0418 |
| type |  | 算法运行环境 | string |  | spark、gbase | 算法运行环境，取值：spark、sybase、gbase等。  说明：1）每个任务只支持单一运行环境，任务内不能同时涉及两个及以上运行环境，比如spark清洗表，表又同步到gbase，这种任务要拆成两个任务 | 6.15.50.03-0418 |
|  | parameter | 自定义参数 | string |  |  | 一般不需要填写 | 6.15.50.03-0418 |
| executecycle |  | 执行周期 | string | province,day:按省每天执行一次  province,week：按省每周执行一次  province,month：按省每月执行一次 | province,week | 任务执行的周期 | 6.15.50.03-0418 |
| executestart | 算法执行周期的开始时间 | int | Week:[1,7]  Month：[1,自然月末] | 1 | 缺省值：空，表示每天执行；取值为整数，若为周任务，取值为1到7，月任务取值为1到30或28或31，执行周期是day，hour任务时不需要填写 | 6.15.50.03-0418 |
| datastart | 依赖的数据开始时间 | int | Week:[1,7]  Month:[1,自然月末] | 1 | 若执行的是周一到周日的任务，则取值为1；执行周期是day和hour，则无需填写 | 6.15.50.03-0418 |
| dataend | 依赖的数据结束时间 | int | Week:[1,7]  Month:[1,自然月末] | 7 | 若执行的是周一到周日的任务，则取值为7；执行周期是day和hour，则无需填写；如果依赖的数据是月末结束，则无需填写，因为存在是否30,31的情况 | 6.15.50.03-0418 |
| cycle |  |  |  |  |  | 6.15.50.03-0418 |
| cron | 定时执行时间 |  |  | 0 37 8 \* \* ? \* | 配置 cron时，同时需将[任务XML文件说明](#_任务XML文件说明)中的type修改为timedriver；如0 37 8 \* \* ? \*表示每天的8点37分执行 | 6.15.50.03-0418 |
| checkpointrecovery | 是否在上电时生成该时间点以前的当天所有定时任务 |  |  | true/false | 作用： 是否在上电时生成该时间点以前的当天所有定时任务。  默认不填为ture即生成所有任务，填为false的话只生成该上电时间点以后的定时任务 | 6.15.60.03-0907 |
| scenesetting | type | 场景设置 |  |  |  | 除了核心网，其他的不配置或者默认为空 | 6.15.50.03-0418 |
| timesetting | 忙闲时设置 | String | busymorningtime，busynighttime，idletime，usertabletime | busymorningtime，busynighttime | 取值：busymorningtime，busynighttime，idletime，usertabletime是用于配置某一张表的忙闲时字段，和timesetting.conf中的字段对应即可。支持忙闲时任一组合，不填默认没有,例子见 | 6.15.50.03-0418 |
| algorithminfo | subtype | 算法类型 | string | rdd、sql、rddsybase、rddgbase、webgisfile、sybasefile、hbasefile、gbasefile  gbaseload、  gbasejar、  gbasesql、 fileupload | rdd | 算法为纯的spark jar包时填写rdd；  算法为spark运行指定sql文件时填写sql；算法为spark环境运行同时操作gbase的jar包时填写rddgbase；  算法为spark环境运行同时操作sybase的jar包时填写rddsybase；  算法为生成webgis文件时填写webgisfile；  算法为sybase数据入库时填写sybasefile； gbasefile，spark数据同步至gbase，与gbase数据入库任务独立；  算法为hbase数据入库时填写hbasefile；  gbasesql：纯gbase任务的sql类型，sql语句中只支持 begintime和endtime标签，为含时分秒的时戳，即用$表名.begintime$作为条件进行筛选；  gbasejar，纯gbase任务的jar类型；  gbaseload，spark任务类型，输入是spark表，输出是gbase表，相当于gbasefile类型，但同时包含导出文件和入库gbase两个步骤； fileupload，算法输出结果为.csv文件形式，并将文件推送到 max-metadata-manager/metadata/config/dashboardFtpPath.conf指定的ftp服务器指定目录下 | 6.15.50.03-0418、V6.15.60.02支持 fileupload |
| classname | 算法jar包中类名信息 | string |  | xxx | Subtype为rdd，rddgbase，rddsybase，gbasejar时必填，指定jar包类名 | 6.15.50.03-0418 |
| classmethod | 算法jar包中类的方法名 | string |  | action | Subtype为rdd，rddgbase，rddsybase， gbasejar时必填，指定jar包类型 | 6.15.50.03-0418 |
| jarpath | 算法jar包路径 | string |  | D:\algorithm\umts\_subject\_coverage\_grid\_week\algorithm.jar | Subtype为rdd，rddgbase，rddsybase， gbasejar时必填，指定jar包路径 | 6.15.50.03-0418 |
| sqlpath | 算法sql文件路径 | string |  |  | Sql、 gbasesql写此项，其他不填 | 6.15.50.03-0418 |
|  | confpath | 算法配置文件路径 | string |  |  | Sql或jar包有配置参数时必填 | 6.15.50.03-0418 |
| inputparameter | parametername | 算法输入表别名 | string |  | Tableprojdata | 算法中依赖表的别名或其他依赖别名，示例如下： 原select \* from umts\_cm\_projdata，把umts\_cm\_projdata取一个别名Tableprojdata，此时sql写为select \* from Tableprojdata，而parametername取值为Tableprojdata | 6.15.50.03-0418 |
| value | 算法输入表取值 | string |  | umts\_cm\_projdata | 算法中依赖表或其他依赖的真实名字 | 6.15.50.03-0418 |
| type | 算法输入参数数据库类型 | string | Spark、sybase、gbase、hbase、  spark\_cache  spark\_txt\_file  spark\_xml\_file  spark\_ems\_file | spark | 已确定需要到达检查的数据表则填写spark\_xml\_file 或 spark\_txt\_file或spark\_ems\_file；不需要到达检查的数据表则填写为表所属数据库类型，如spark或sybase或gbase或hbase | 6.15.50.03-0418 |
| dbname | 算法输入参数数据库名称 | string |  | zxvmax | 表数据库名 | 6.15.50.03-0418 |
| partition | 算法输入表分区 | string |  | p\_date | 1)与建表顺序一致，用到哪级则顺序填写到哪级,如分区包括省、月、天、小时，但算法只用天分区，则需要填写省、月、天作为入参 2)分区取值要与sql文件中的分区变量名一致 3）表的数据到达粒度和表的纬度等信息由表元数据提供 | 6.15.50.03-0418 |
| outputparameter | parametername | 算法输出参数名 | string |  | CoverageSubjectGridDa y | 算法输出表的别名，sql中需要使用这个别名，不能使用真实的表名 | 6.15.50.03-0418 |
| value | 算法输出参数取值 | string |  | umts\_subject\_coverage\_grid\_day | 算法输出表真实名字 | 6.15.50.03-0418 |
| type | 算法输出参数数据库类型 | string | Spark、sybase、gbase、  spark\_cache  spark\_txt\_file  spark\_xml\_file | spark | 表数据库类型 取值为spark，sybase，hbase等；对于dataStorage 入库前的文件的tableType 填 spark\_xml\_file 和 spark\_txt\_file；spark\_cache：spark缓存表 | 6.15.50.03-0418 |
| dbname | 算法输出参数数据库名称 | string |  | zxvmax | 表数据库名 | 6.15.50.03-0418 |
| selfdependence | 是否自依赖 | string |  | TRUE | 取值为false，或true。缺省为false，当sql中的表存在既是输入表又是输出表时，需将该字段设置为true。例如，输入昨天的周表和天表输出今天的周表时，对应的周表的该字段需设置为true | 6.15.50.03-0418 |
|  | partition | 算法输出表分区 | string |  | p\_date | 输出表分区要与建表脚本一致，需填写输出表的所有分区 | 6.15.50.03-0418 |
| resource |  | 算法所需资源（核数、内存等） | List[corenum,memory] | 0~100 | List[50,50] | 单位为%，占用系统资源的百分比，建议单个任务的资源配置不大于20%；配置过大时，易对系统资源造成独占，且在其他任务运行时剩余资源不足配置时，会出现因资源不足不易被调度 | 6.15.50.03-0418 |
| deleteinfo | deletetype | 删除类型 | string |  | autodelete | 可以填写两种类型assistdelete（补踩数据时，需要调度删除数据）和autodelete（默认和不填写时，算法自己删除） | 6.15.50.03-0418 |
| spark\_special\_parameter |  | 用于算法中单独调整spark的hc和sc参数(分别指以hc.sql(set xxx)和 sc.set()方式设置的参数） | String | vmaxs\_parameter\_codegen.sql，vmaxs\_parameter\_codegen\_forindividual.sql，vmaxs\_parameter\_default.sql，vmaxs\_parameter\_DynamicPartition.sql，vmaxs\_parameter\_ran\_default.sql； | spark\_context\_codegen.sql | 算法需要自行调整hc、sc等参数时，此处配置相关参数的文件名；文件名格式为\*.sql，存放在vmax-etl\config目录下，安装后系统自动同步至目录/metadata/config  1、元数据已提供一些常见sc和hc参数设置的配置文件，业务可根据需求选择使用，配置参数文件名如下：  vmaxs\_parameter\_codegen.sql，vmaxs\_parameter\_codegen\_forindividual.sql，vmaxs\_parameter\_default.sql，vmaxs\_parameter\_DynamicPartition.sql，vmaxs\_parameter\_ran\_default.sql；  2、如上述配置文件无法满足需求时，业务可自行在vmax-etl\config目录下增加\*.sql的配置文件，要求如下：  1）特殊SC或HC参数的配置文件必须以“spark\_context\_”开头，结尾可接业务名等，如spark\_context\_lte\_calldrop.sql  2）其他特殊提交参数，会连接在json串最后直接提交，配置参数文件必须以“submit\_params\_”开头，可设置的参数如执行核提交--executor-cores 2等  3） spark\_context\_和 submit\_params\_开头的参数文件可同时配置，以”,”分隔  4）不能同时配置多个 spark\_context\_或 多个submit\_params\_；  5）若新增\*.sql文件命名规范不满足上述要求时，需在metadata-dataprocess.conf文件中增加标签进行sql文件的映射配置，标签需与新增\*.sql文件名一致 | V6.15.60.02之前支持方法1， V6.15.60.02增加方法2， |
| Iscache |  | 算法内部是否cache | int |  | 0 | 做了cache操作，取值为1，否则取值为0 | 6.15.50.03-0418 |
| prefield\_1 |  | 预留字段\_1 |  |  |  |  | 6.15.50.03-0418 |
| prefield\_2 |  | 预留字段\_2 |  |  |  |  | 6.15.50.03-0418 |
| prefield\_3 |  | 预留字段\_3 |  |  |  |  | 6.15.50.03-0418 |

#### 示例说明

Xml中输入和输出表名的用法：将相应的算法sql中的输入输出表名，输入表分区名字提取出来，替换为xml中相应表的别名，分区替换为$表别名.分区$。例如输入表cdma\_subject\_weakcover\_grid\_1x\_day的别名为gridcoverDay，分区名为'$gridcoverDay.p\_date$'，用法见章节[Sql文件说明](#_Sql文件说明)。

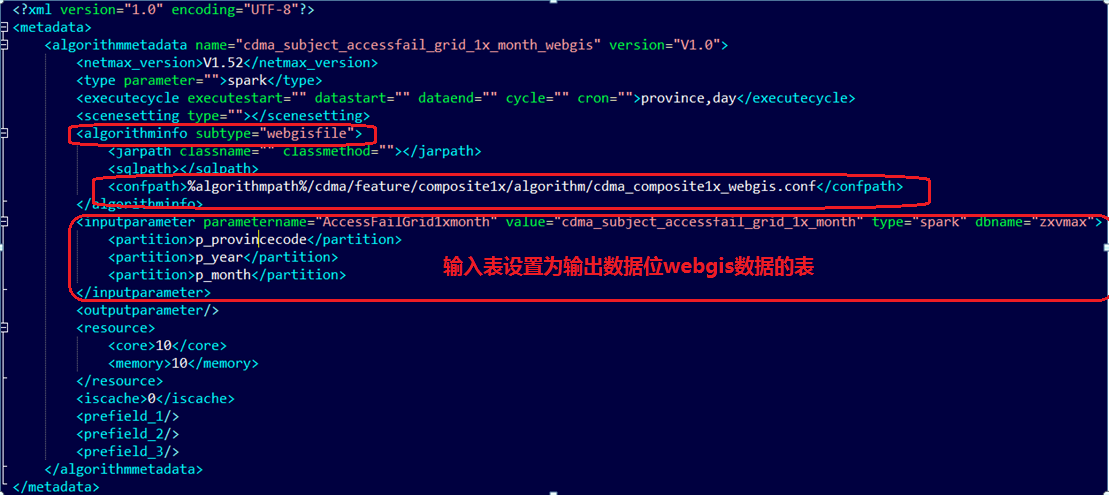


### **WebGis XML文件说明**

#### 字段说明

对于需生成webgis预处理文件的算法，需配置\*\_webgis.xml。字段说明同算法xml文件，只需将algorithminfo.Subtype算法类型配置为webgisfile类型，并在confpath中指明webgis预处理文件相关的conf文件。

#### 示例说明



### **Gbase XML文件说明**

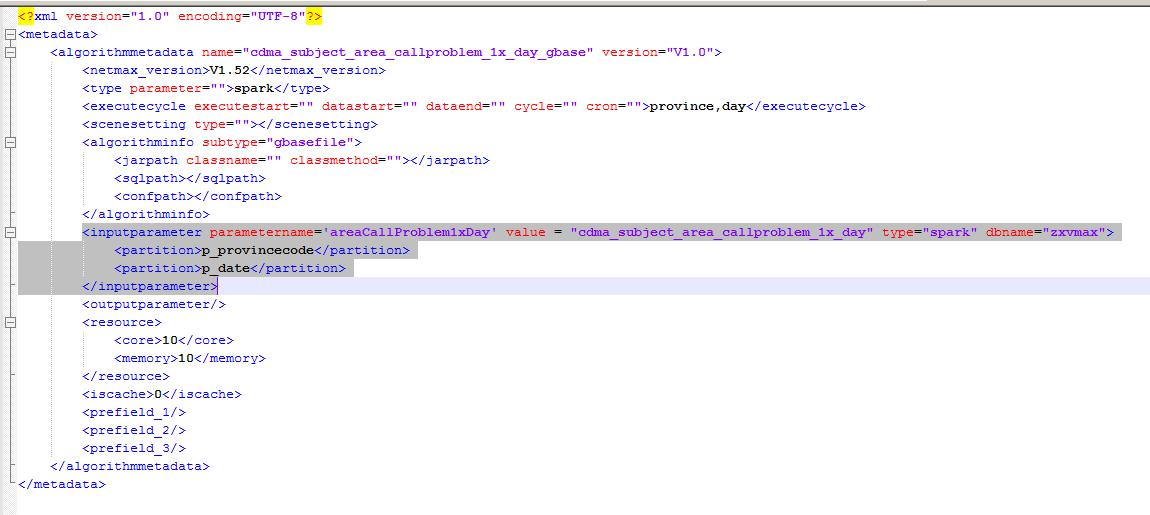
#### 字段说明

对于需要从spark上同步到gbase的表，由平台统一完成，应用需配置\*\_gbase.xml文件。字段说明同算法xml文件，只需将algorithminfo. Subtype算法类型配置为gbasefile类型。

缺省情况下，spark和gbase上的表名完全一样，此时无需配置输出表名。对于spark和gbase表名不一致的情况，需要将同步算法的xml中增加输出表的表名，并且输出参数中的类型改成gbase，例如：

< outputparameter parametername='areaCallProblem1xDay' value = "cdma\_subject\_area\_callproblem\_1x\_day" type="gbase" dbname="zxvmax">

#### 示例说明



### **任务XML文件说明**

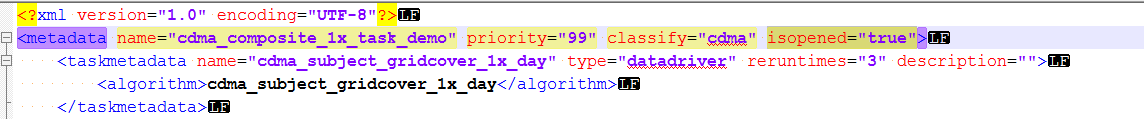
1.svn存放路径：\networkoptimization\cdma\feature\XXXX\task下，按照制式分目录，此处举例为cdma，可以是LTE，UMTS等其他制式；XXXX具体的特性应用功能名，如composite1x。

2.文件名命名规则：制式\_功能\_task.xml；

#### 字段说明

| **节点** | **属性** | **中文** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| metadata | name | 虚拟任务名 | string |  | umts\_coverage\_overshoot\_task | 虚拟任务包含了多个任务，虚拟任务名尽可能准确描述这个xml的业务 | 6.15.50.03-0418 |
| priority | 优先级 | Int |  | 99 | 数值越大，优先级越高 | 6.15.50.03-0418 |
| classify | 制式名称 | String | umts、lte、cdma | lte | 取值为lte，cdma，umts等 | 6.15.50.03-0418 |
| isopened | 虚拟任务开关 | String |  | isopened="true" | 取值为true和false，分别表示虚拟任务的开和关，默认不填代表true | 6.15.50.03-0418 |
| taskmetadata | name | 任务名 | string |  | umts\_subject\_coverage\_cell\_month |  | 6.15.50.03-0418 |
| type | 任务类型 | string | datadriver、timedriver、runimmediate、userdriver | datadriver | datadriver:表示数据驱动任务，任务是否执行由所依赖表的数据公告牌状态决定；timedriver用于定时任务，任务按照配置的时间定时执行；  runimmediate质量检查任务，暂不使用；userdriver：用于用户驱动任务，任务在用户触发后才会上牌执行； | 6.15.50.03-0418 |
| reruntimes | 重试次数 | int |  | 3 | 失败重试次数 | 6.15.50.03-0418 |
| description | 任务描述 | string |  | 覆盖分析小区月统计 |  | 6.15.50.03-0418 |
| priority | 任务优先级 | Int |  | 99 | 默认没有该属性，采用虚拟任务的优先级，如果填写了，则以任务优先级为准。 |  |
| algorithm |  | 算法名称 | string |  | umts\_subject\_coverage\_cell\_month | 需要和算法xml里面的算法名保持一致 | 6.15.50.03-0418 |

#### 示例说明



### **表XML文件说明**

1.svn存放路径：\vmax-etl\originaltable下，按照数据库类型分目录，例如，分为spark，sybase、gbase目录等；目录下可以按照制式区分文件；

2.文件名命名规则：tab\_制式\_数据源.xml；目前的定义方式是：对于netmax原始数据源主题表，会命名为tab\_制式\_netmax的方式，其他则无该标识。

3.每个表一个tablemetadata 记录，每个field代表一个表字段信息。

#### **字段说明**

| **节点** | **属性** | **中文** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| metadata |  |  |  |  |  |  | 6.15.50.03-0418 |
| tablemetadata | name | 节点名称，也就是表名 | string |  | umts\_subject\_coverage\_weakcover\_cell\_day | 需要和汇总表中的表名保持一致 | 6.15.50.03-0418 |
| tabletype | 表类型 | string | Spark、sybase、gbase、gbase\_dynamic | sybase | 表的数据库类型；gbase\_dynamic目前仅用于核心网分表情况 | 6.15.50.03-0418  V6.15.60.03B3支持gbase\_dynamic |
| field | Fieldid | 字段名 | string |  | 如字段修改示例：  源表字段：<field fieldid ="days" fieldname="days" fieldtype="int".../>  修改后字段：<field fieldid ="days" fieldname="dayByDay" fieldtype="string".../> | 用于表升级和动态修改表字段。  分为字段修改，新增字段，删除字段三种操作；spark上不支持删除字段。  新增字段时，新增一条<field/>，其中fieldid与fieldname相同；  字段修改时，修改原<field/>，其中fieldid="旧字段名称"，fieldname="新增字段名称"。  删除字段时，删除对应的<field/> | 6.15.50.03-0418 |
| fieldname | 字段名 | string |  | TOP1AccessFailReasonName | 表字段，字段名称不能为空 | 6.15.50.03-0418 |
| fieldtype | 字段类型 | string |  | Varchar(80) | 有int、double、string、varchar(80)、bigint等，不能填空；字段类型需与相应的数据库要求一致 | 6.15.50.03-0418 |
| notnull | 字段是否不允许为空 | int | 0、1 | 0 | 1表示不允许，0表示允许 | 6.15.50.03-0418 |
| length | 长度 | string |  | 255 | 如果没有长度则不填；如果是有精度信息的则用“ 12,2”这种格式填写 | 6.15.50.03-0418 |
| isprimarykey | 是否主键 | int | 0、1 | 0 | 是填写1；不是填写0 | 6.15.50.03-0418 |
| description | 描述 | string |  | 出现最多的接入失败原因名称 | 缺省为空 | 6.15.50.03-0418 |
| algorithm | 算法 | string |  |  | 计算字段的方法，缺省为空 | 6.15.50.03-0418 |
| source | 来源 | string |  |  | 缺省为空 | 6.15.50.03-0418 |

#### **示例说明**



### **汇总**表xml说明

1、svn存放路径：\vmax-etl\originaltable下，按照数据库类型分目录，例如，分为spark，sybase目录等；目录下可以按照制式区分文件；

2、文件名命名规则：summary\_tab\_制式\_数据源.xml；目前的定义方式是：对于netmax原始数据源主题表，会命名为summary\_tab\_制式\_netmax的方式，其他则无该标识。

3、每个field节点表示一个表的信息

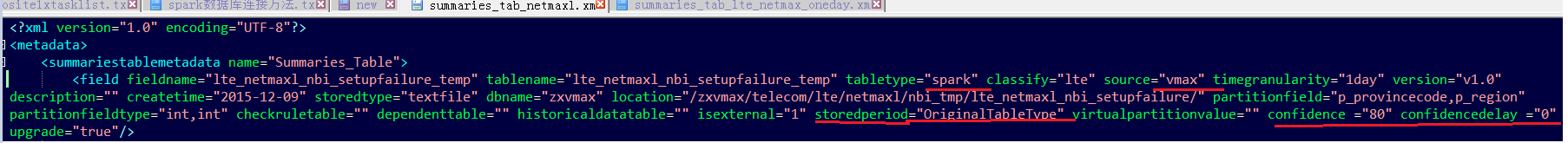
#### **字段说明**

特别注明：1. 所有汇总表中（spark、sybase、gbase、spark\_txt\_file等），classify字段用于制式控制，必须填写正确，否则可能导致任务上不了公告牌。

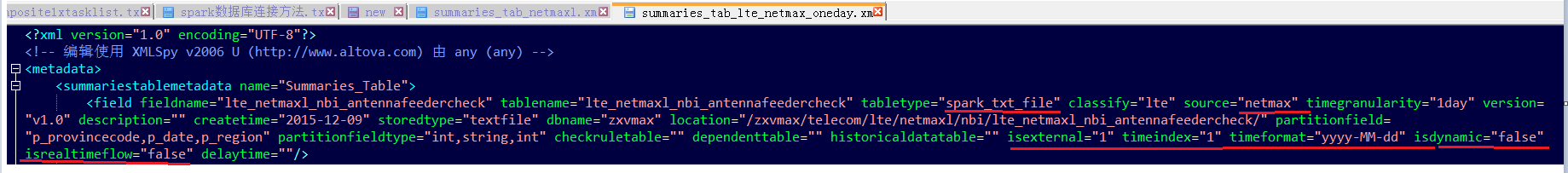
2. 2016年6月22号，删除etl主干上一直没有使用的checkruletable、dependenttable、historicaldatatable三个字段。

| **节点** | **属性** | **中文** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| metadata |  |  |  |  |  |  | 6.15.50.03-0418 |
| summariestablemetadata | name | 节点名称 | string |  | Umts\_Summaries\_Table |  | 6.15.50.03-0418 |
| field | fieldname | 节点名称 | string |  | umts\_subject\_coverage\_weakcover\_cell\_day | 目前和tablename保持一致 | 6.15.50.03-0418 |
| tablename | 具体表名 | string |  | umts\_subject\_coverage\_weakcover\_cell\_day | 后续可能表名都带版本号 | 6.15.50.03-0418 |
| tabletype | 表类型 | string | spark、sybase、gbase  、netmax、gbase\_dynamic | sybase | 需要做数据到达检查的表，统一配置为netmax，否则配置为数据库类型，如spark或sybase或gbase或hbase等；gbase\_dynamic目前仅用于核心网分表情况 | 6.15.50.03-0418 V6.15.60.03B3支持gbase\_dynamic |
| classify | 归类 | string | lte、cdma、umts，union、cn、cn-cdma、cn-lte、cn-union、cn-umts、COMMON、userprofile , fixednetwork,serviceexperience,iptv,multinetwork,health | umts | 算法所依赖的数据源制式，需根据实际填写，并与vmax-conf/datasourceconfig.xml配置中该省市的制式一致；该字段会涉及制式控制，必须填写正确，否则可能导致任务上不了公告牌。 | 6.15.50.03-0418、V6.15.60.01B2版本支持fixednetwork,serviceexperience,iptv,multinetwork,health |
| source | 来源 | string | Netmax，vmax，dcu | coverage | cn侧的数据源表配置为 dcu，cn侧清洗出的表配置为vmax；  ran侧需要到达检查的netmax数据表则配置为netmax ，中间清洗数据表需要到达检查配置为vmax；  工参、字典表、定时任务等没有输出若需要到达检查则配置为netmax； | 6.15.50.03-0418 |
| timegranularity | 生成时间粒度 | string | 1hour、1day、1week、1month | 1day | 1hour、1day、1week或者1month | 6.15.50.03-0418 |
| version | 版本号 | string |  | v1.0 | 表的版本号 | 6.15.50.03-0418 |
| description | 描述 | string |  | 弱覆盖小区天表 |  | 6.15.50.03-0418 |
| createtime | 创建时间 | string |  | 2015/11/21 | 表创建时间 | 6.15.50.03-0418 |
| storedtype | 数据库文件格式 | string | ORC、TEXTFILE、parquet | ORC | ORC、TEXTFILE、parquet | 6.15.50.03-0418 |
| dbname | 数据库名 | string | zxvmax | zxvmax | 所在数据库名 | 6.15.50.03-0418 |
| location | 表文件所在路径 | string |  | /zxvmax/telecom/cdma/subject/cdma\_netmaxc\_subject\_tenthousand\_user\_calldrop\_month/ | spark表才有，sybase不需要；对于原始数据入库的表需填写为drs推送到hdfs的路径（如/cdma/netmaxc/ndsloader\_tmp/callmodel/cdma\_netmaxc\_nbi\_callmodel\_call\_model\_by\_serv），细化需要动态分区的小时表填写为临时路径，其他表填写正式路径（/cdma/ems/cdma\_cm\_neighborcell\_config\_do/ | 6.15.50.03-0418 |
| partitionfield | Spark表配置中表示表分区；sybae/gbase表配置中表示从spark同步至sybase/gbase时使用的表字段名 | string |  | p\_provincecode,p\_date | 多个分区的话按照英文逗号分隔，顺序就是创建表的顺序；由于导出成sybase/gbase文件后，sybase/gbase入库之前需要删除重复数据。如果sybase/gbase表的时间字段不是标准的，需要在sybase/gbase汇总表中填写时间的字段信息，目前利用汇总表中的两个属性partitionfield和partitionfieldtype来填写。例如，如果是周和月的表有多个时间信息，则用逗号分隔 | 6.15.50.03-0418 |
| partitionfieldtype | spark表分区字段类型sybae/gbase表配置中表示从spark同步至sybase/gbase时使用的表字段类型 | string |  | int,string | 多个分区的话按照英文逗号分隔，和表分区partitionfield字段顺序对应 | 6.15.50.03-0418 |
| isexternal | 是否外部表 | Int | 0,1 | 0 | 0表示内部表，1表示外部表；支持平台中的动态修改。 | 6.15.50.03-0418 |
| storedperiod | 数据存储周期,单位为天 | string | GbaseDefaultType, SparkDefaultType , SparkNetmaxType, | SparkDefaultType | 缺省为空；在dataperiodmanage.conf中配置DataStoredPeriod类型的保存时间周期，配置500就是按当天时间保存500天数据，该参数可供整个系统使用。  GbaseDefaultType是gbase默认保留天数;SparkDefaultType是spark默认保留天数;SparkNetmaxType是spark的netmax表保留天数.  DataStoredPeriod{  GbaseDefaultType = 500  SparkDefaultType = 500  SparkNetmaxType = 500  }  详见章节5.4[数据生命周期配置](#_数据生命周期配置) | 6.15.50.03-0418 |
| virtualpartitionvalue | 虚拟分区 | string | virtualcdma，virtuallte | virtualcdma | 缺省为空；表配置虚拟分区表示该表同一分区数据来源与多次清洗，清洗次数和虚拟分区的个数一致。目前只有核心网使用该字段 | 6.15.50.03-0418 |
| timeindex | 表示文件中表示时间信息字段开始的位置， | Int |  | 1 | 前提tableType或datasourcetype配置为spark\_txt\_file，则该字段需要配置，取值从1开始，若tableType配置为非spark\_txt\_file，则该字段可以不配置 | 6.15.50.03-0418 |
| timeformat | 时间字段的format格式 | String |  | yyyy-MM-dd HH:mm:ss | 前提tableType或datasourcetype配置为spark\_txt\_file，则该字段需要配置，但是格式取决于文件内时间格式 | 6.15.50.03-0418 |
| isdynamic | 是否使用动态分区的标识 | 布尔 | True，false | false | 使用动态分区表的isDynamic 字段需要填true，不填使用默认值，默认值为false；dataStorage 任务中，小时表的动态分区的 isDynamic 字段需要填 true | 6.15.50.03-0418 |
| isrealtimeflow | 表示是否为实时流， | 布尔 | True，false | false | 给cn用，不填使用默认值，默认值为false | 6.15.50.03-0418 |
| delaytime | 表示延迟翻牌的时间，单位为分钟 | Int |  |  | 检查文件到达后，延迟翻牌的时间，时间粒度为分钟，缺省为空，不填使用默认值 | 6.15.50.03-0418 |
| confidence | 表的置信度 % | Int | 0~100 | 80 | 表的置信度，百分比形式 | 6.15.50.03-0418 |
| confidencedelay | 延迟检查时间 | Int |  | 1 | 如果数据未全部达到，且到达个数超过置信度，延迟小时/天之后有数据达到，则任务可以执行；粒度和任务相同。  如天表依赖小时表的清洗任务，confidence = 80，confidencedelay = 0 表示：依赖的小时表分区数量 >= 24\*80%，即大于19.2,即可执行  天表依赖小时表的清洗任务，confidence = 80，confidencedelay = 1 表示：依赖的小时表分区数量 >= 24\*80%，即大于19.2,且 到下一个分区（小时）到来之后可执行  天表依赖小时表的清洗任务，confidence = 80，confidencedelay = 2 表示：依赖的小时表分区数量 >= 24\*80%，即大于19.2,且 到下2个分区（小时）到来之后 可执行  月表依赖天表的清洗任务，confidence = 80，confidencedelay = 1 表示：依赖的天表分区数量 >= 当月天数\*80%，且 到下一个分区（天）到来之后 可执行 | 6.15.50.03-0418 |
| upgrade | 表是否可以升级 | 布尔 | True，false | True | "upgrade='true'" 表示表可以支持升级。"upgrade='false'" 表示表不支持升级。默认不填，表示表可以升级。 | 6.15.50.03-0418 |
| datasourcetype | 数据到达检查策略方式 | String | dcu、spark\_partition、spark\_txt\_file、spark\_xml\_file、spark\_ems\_file、noDynamic、gbase\_import、broadband、iptv、streaming\_partition | noDynamic | Dcu：Cn自行实现检查  spark\_partition：检查表分区  spark\_txt\_file：检查文件内容  spark\_xml\_file：检查文件个数  spark\_ems\_file：不检查  noDynamic：检查文件是否存在  gbase\_import：检查gbase中是否有数据；  broadband : 宽带数据到达检查，检查文件目录，不含数据中心分区(目录结构类似/xx/yy/zz/ p\_provincecode=5100/p\_date=2016-06-06,判断准则为某省当天目录下有文件且该省存在之后时间目录，如p\_provincecode=5100若想翻2016-06-06号的牌，必须满足该天目录下有文件，而且2016-06-06之后的目录也存在，比如2016-06-07、2016-06-08等)  iptv : iptv数据到达检查，检查文件目录，含数据中心分区(目录结构类似 /xx/yy/zz/p\_provincecode= 5100/p\_datacenter=22/p\_date=2016-06-06判断准则为某省的某数据中心当天目录下有文件且该数据中心存在之后时间目录，如p\_provincecode=5100、p\_datacenter =22数据中心若想翻2016-06-06号的牌，必须满足该天目录下有文件，而且2016-06-06之后的目录也存在，比如2016-06-07、2016-06-08等)  streaming\_partition：streaming检查方式，检查该分区及该分区之后某时间分区是否存在  详细见章节5.3[数据到达检查相关配置](#_数据到达检查相关配置) | V6.15.60.01B5、V6.15.60.03B1增加streaming\_partition |
| fieldsterminatedby | Spark表的数据列分隔符 | String | 默认不填该列，用逗号分隔；如果填写了，以填写的字符分隔。 | | | 目前是有数据用|分隔的，类似这种特殊的数据，需要填写该字段 | V6.15.60.02B1 |
|  | tableoption | 建表脚本里面一些特殊的选项 | String | 默认不填该列，如果建表脚本里面一些特殊的选项，比如DISTRIBUTED BY('id')，则可以填写 | DISTRIBUTED BY('id') | 如果填写了值，则会在sql建表脚本里面增加相应的选项，比如:  create table report\_windowsip(    id int auto\_incrementnot null primary key,    ip varchar(20),    time timestamp )**DISTRIBUTED BY('id')**; | 6.15.70.01B1 |

#### **主题表示例说明**



#### **Datastorage入库表示例说明**



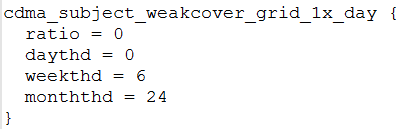
## Conf文件说明

应用开发需配置的有两种conf文件，算法conf文件和weggis预处理文件的conf文件配置。两种conf文件说明如下。

### 算法conf说明

算法.conf文件的主要内容为算法中的门限等相关配置，如果算法中的某些门限值是通过配置文件读取得到的，则需要有一个相应的算法.conf文件。内容为算法中的门限值。如ratio:20,说明率ratio的值为20；配置参数在sql.文件中以##标识；如下sql和对应的conf配置为例：





### **Webgis conf说明**

webgisfile的配置文件与其他算法的配置文件内容不同，需要根据算法名字配置相应的渲染数据文件中的列名和相应的sld文件名字。如下，需配置webgisinfo项，有多个表的配置时以如下形式配置：

表名{

webgiscolumns=“……” //数据来源字段

webgistitlecolumns=“……”//webgis要求的接口字段

webgissldfilename=“……”//sld文件名

}

示例如下:



## Sql文件说明

Sql文件是算法逻辑实现的表现形式。需配合[算法XML文件说明](#_算法XML文件说明)和[算法conf说明](#_算法conf说明)共同实现算法逻辑。

### 文件编码格式要求

Sql文件的编码格式不能选择UTF-8(无BOM格式编码)格式，否则sql中包含中文时，有可能会出现算法异常；建议统一采用ANSI编码格式。

### 替换符号

Sql文件中，有两种符合被用作关键字“$”和“#”。其中“$”用于表示算法xml中的表别名，表分区等信息，在sql中用“$表别名$”标识；“#”用于标识算法conf中的配置参数，在sql中通过“#门限#”标识。

例如输入表cdma\_subject\_weakcover\_grid\_1x\_day的在算法xml中的别名为gridcoverDay，分区名为'$gridcoverDay.p\_date$'。将算法sql中的门限值替换为对应conf文件中的配置值例如ratio，sql文件中的形式则为#ratio#，具体示例如下：



### 替换规则

sql类型实际在运行时也是打成jar包的，只是打jar包的操作由平台统一实现了。jar包内会调用sql，把实际sql中的参数替换。jar包中传递给sql中的信息包括：输入表名、输出表名、输入表分区、输出表分区。

在实现sql文件时，元数据平台提供了以下原则和实用接口：

1）sql中的分区，输入表使用输入表分区，输出表使用输出表分区，分区等需使用$表别名.分区$的形式；

2）若输入、输出表有p\_date分区，则jar包会计算出如下时间共算法使用

| **时间类型** | **说明** | **示例** |
| --- | --- | --- |
| p\_year =2015 | 若有p\_date分区，则有此参数，年 | $表别名. p\_year $ |
| p\_month =11 | 若有p\_date分区，则有此参数，月 | $表别名. p\_month $ |
| p\_week =48 | 若有p\_date分区，则有此参数，周 | $表别名. p\_week $ |
| p\_date=2015-11-30 | 天，sql直接使用，不要再做加减运算 | $表别名. p\_date $ |
| p\_date\_min=2015-11-30 | 每周、每月执行一次的任务的，时间x<p\_date<y，x= p\_date\_min； | $表别名. p\_date\_min$ |
| p\_date\_max=2015-12-03 | 每周、每月执行一次的任务的，时间x<p\_date<y，y= p\_date\_max， | $表别名. p\_date\_max $ |
| firstDayOfWeek=2015-11-30 | 若有p\_date分区，则有此参数，p\_date对应周的第一天（周一）； | $表别名. firstDayOfWeek $ |
| lastDayOfWeek=2015-12-06 | 若有p\_date分区，则有此参数，p\_date对应周的最后一天（周日）； | $表别名. lastDayOfWeek $ |
| firstDayOfMonth=2015-11-01 | 若有p\_date分区，则有此参数，p\_date对应月的第一天（1号）； | $表别名. firstDayOfMonth $ |
| lastDayOfMonth=2015-11-30 | 若有p\_date分区，则有此参数，p\_date对应月的最后一天； | $表别名. lastDayOfMonth $ |
| yearOfWeek=2015 | 根据天表计算出的周表的年； | $表别名.  yearOfWeek $ |
| latestDayOfPartition=2015-11-30 | 根据表天分区计算最近分区（表分区支持day, p\_date, reportdate, date），V6.15.60.03开始支持 | $表别名.  latestDayOfPartition $ |

3）现在的周表、月表的清洗大部分都是累加的，算法xml把天表作为入参，周表不作为入参填写，周表作为出参填写，那么调度调用时，会把入参和出参都带给sql，具体使用分区字段如下：（跨年周的问题也可以解决）：

insert overwrite 周表（p\_year=周表.年分区, p\_week=周表.周分区，p\_date=周表.天分区）

输出参数中周表天分区直接用，不要加减

select ... from 天表 p\_date=天表.天分区

输入参数中天表天分区直接用，不要加减

union all select ... from 周表 p\_year =周表.年分区 and p\_week=周表.周分区 and p\_date=周表.天分区 – 1

输出参数中周表天分区减1

## Rdd jar包说明

对于使用sql文件无法完成的算法，可使用RDD形式，将算法打成jar包形式，提供相应的Action入口给元数据平台。

处理清洗的代码放在一个独立的Module中并添加依赖Moudel:coordinator。**通过继承coordinator中的ActionBase抽象类获得HiveContext、SparkContext、json入参等来完成清洗算法的RDD处理，代码需要import com.zte.vmax.metadata.{ActionResult, ActionEnv, ActionBase}**。最后将Module生成jar包，需要将依赖的外部jar包剔除，避免jar包过大，通过jar包和相应的算法xml配置成一个清洗的任务。需要将编译出的jar包和算法xml等文件放置在同一目录下，目前是对应功能的algorithm目录下。

元数据平台提供了一个统一获取任务信息（json文件）的接口，接口存在于：metadata-coordinator\_1.1.10-SNAPSHOT.jar包；coordinator工程获取路径：[CODE/dev/ZXVMAX/vmax-metadata/vmax-metadata-spark-coordinator](http://10.5.70.3/ZXVMAX/CODE/dev/ZXVMAX/vmax-metadata/vmax-metadata-spark-coordinator)；其中提供了json中的相关信息，包括输入输出表名，配置参数、分区信息等，详细见接口说明。

### 接口说明

**接口名：val params = env.getJSONParams()**

**作用：**统一获取Json任务信息的接口

**返回值：params 说明**

params.algorithmName：获取算法名

params.configPath：获取算法配置文件路径

params.date：获取定时任务的时间

params.dbName：获取数据库名称

params.dbType：获取数据库类型

params.provinceCode：获取省分区

params.sparkConfigPath：获取spark配置文件路径

params.sqlPath：获取sql文件路径

params.taskKey：获取taskKey

params.gbaseInfo：获取gbase连接的相关信息

params.sybaseInfo：获取gbase连接的相关信息

params.inputTable：获取输入表相关信息

params.outputTable：获取输出表相关信息

其中：

1）gbaseInfo、sybaseInfo为case类（DBInfo）类型：

DBInfo(url : String, user : String, password : String, driver : String, dbName : String)

url：数据库连接的url

user：用户名

password：密码

driver：驱动

dbName：database名称

获取url的示例：

val gbaseInfo = params. gbaseInfo

gbaseInfo.url

2）inputTable、outputTable为List[Table]类型，Table为case类：

Table(tableName : String, tableAlias : String, partitions : List[KeySplitResult])

tableName：表名称

tableAlias：表别名

partitions：分区

KeySplitResult (day : String, tblName : String, partitions : List[String], tblType : String ="")

day：公告牌时间

tblName：表名称

partitions：详细分区

tblType：数据库类型

**json内容示例**：

"inputTables": "gb1=test\_gbase1,gb2=test\_gbase2",

"inputPartitions":

["2016-04-18#test\_gbase1#provincecode=510000,startTime=2016-04-18 00:00:00,endTime=2016-04-18 23:59:59#gbase",

"2016-04-18#test\_gbase1#provincecode=510000,startTime=2016-04-19 00:00:00,endTime=2016-04-19 23:59:59#gbase",

"2016-04-18#test\_gbase2#provincecode=510000,startTime=2016-04-18 00:00:00,endTime=2016-04-18 23:59:59#gbase"],

**json内容中表信息解析结果说明**：

val inputInfo = **params.inputTable**

inputInfo.map(\_.**tableName**) /*/List(test\_gbase1, test\_gbase2)*

inputInfo.map(\_.**tableAlias**) // *List(gb1,gb2)*

inputInfo.map(\_.**partitions**) //***Lis****t(KeySplitResult (2016-04-18, test\_gbase1，provincecode=510000,List(startTime=2016-04-18 00:00:00,endTime=2016-04-18 23:59:59),gbase), KeySplitResult (2016-04-19, test\_gbase1，provincecode=510000,List(startTime=2016-04-19 00:00:00,endTime=2016-04-18 23:59:59),gbase)) ，*

***List****(KeySplitResult (2016-04-18, test\_gbase2，provincecode=510000,List(startTime=2016-04-18 00:00:00,endTime=2016-04-18 23:59:59),gbase))*

### 注意事项

1）rdd中的返回值不要加反斜杠” \ “等特殊字符

由于返回的信息中包含了反斜杠 "\"，此反斜杠与字符串的结束符单引号 "'"构成转义，导致该字符串无法正常结束，导致sql执行失败。如：

new ActionResult(true, s"fireDate ++++ $fireDate\\ninTableName +++++++++++++ $inTableName\\job +++++++++++++ $serviceID++++$level\\execute ++++++++++++$result")

RDD包中，返回值包含了反斜杠（如上），在返回结果的处理中，会对返回结果做一定长度的截取，而且刚好截取到反斜杠所在位置，导致返回字符串的最后一位为反斜杠,从而造成最后的sql执行失败。

# 流任务实现文件说明

数据源经Spark Streaming接入之后，经过如下步骤最终输出到外部表/文件

1) decoder解码，输出DStream

2) 经过到达检查（可选），输出DStream

3) 经过若干步transform操作，输出DStream

4) 经过output操作，输出到外部表/文件

业务实现streaming任务，需要配置任务task xml，算法 XML，rddtable表xml，以及rdd算法jar包的实现。

## 流任务特性结构说明

业务xml和代码路径与[开发实现文件说明](#_开发实现文件说明)离线任务一样，例如新增一个lte的流任务，则在…/vmax-etl/networkoptimization/lte/feature路径下新建一个目录，以流任务名称命名，在该目录下新增如下四个目录：

algorithm：算法xml、算法jar包；

task：流任务xml

project：算法代码工程，主要配置jar包依赖

rdd：算法实现代码

输入、输出过程RDD[表XML文件](#_表XML文件说明_1)每行各个字段的定义xml和[表XML文件说明](#_表XML文件说明)中spark表xml定义一样，放在路径…/vmax-etl/originaltable/spark下面。

## XML文件说明

一个完整的streaming处理流程包含三种xml：任务task XML，算法alrorithm XML，rddtable表XML。以下分别对三种XML配置文件进行说明。

### Task XML文件说明

task为整个流的属性，主要包含4部分内容;

1) 接入的数据源（例如kafka topic）信息；

2）流处理过程中用到的需要cache的外部表；

3）流处理需要占用的spark资源；

4）流处理各个子过程算法名称

#### 字段说明

| **字段名** | **子字段** | **字段说明** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| streamingtask | name | 流任务名称，用于区分不同的流任务 | 字符串 | 无 | realtimelocation |  | V6.15.60.03B3 |
|  | priority | 优先级 | Int |  | 1000 | 建议高于离线任务 | V6.15.60.03B3 |
|  | classify | 制式，用于数据源控制 | 字符串 | te、cdma、umts，union、cn、cn-cdma、cn-lte、cn-union、cn-umts、COMMON、userprofile , fixednetwork,serviceexperience,iptv,multinetwork,health | cdma |  | V6.15.60.03B3 |
|  | batchduration | 批处理间隔，单位秒 | Int |  | 5 |  | V6.15.60.03B3 |
|  | checkPointPath | checkPoint数据保存路径 | 字符串 |  |  | 断点恢复的Hdfs路径，用于数据断点恢复，缺省为空 | V6.15.60.03B3 |
|  | description | 任务描述 | 字符串 |  |  |  | V6.15.60.03B3 |
|  | isopened | 任务是否上线运行 | Bool | true,false | true |  | V6.15.60.03B3 |
| spark\_special\_parameter | spark\_special\_parameter | Spark特殊参数文件路径， | 字符串 |  | spark\_context\_streaming.sql | spark特殊参数的配置和离线任务一样，这里配置特殊参数的名称，假设配置为spark\_context\_streaming，则需要在…/vmax-etl/config下添加spark\_context\_streaming.sql，在spark\_context\_streaming.sql中放入参数名称和参数值，格式为：**set spark.rdd.compress=true;** | V6.15.60.03B3 |
| auxiliary\_operation | class | 与流无关的辅助操作类名 | 字符串 |  |  | 与流无关的辅助操作类名：比如在流创建前、流停止后执行一些清理操作，如果不需要操作不填改值或者填为空 | V6.15.60.03B3 |
|  | jarpath | 与流无关的辅助操作jar包路径 | 字符串 |  |  | 与流无关的辅助操作类名：比如在流创建前、流停止后执行一些清理操作jar包路径，如果不需要操作不填改值或者填为空 | V6.15.60.03B3 |
|  | confpath | 流任务配置信息路径 | 字符串 |  |  | 配置形式同离线任务，无配置填为空 | V6.15.60.03B3 |
| datasource | name | 数据源名称，用于区分不同的数据源 | 字符串 |  | kafkademo |  | V6.15.60.03B3 |
|  | type | 数据源类型，kafka、socket等 | 字符串 | kafka、socket | kafka | 目前只支持kafka | V6.15.60.03B3 |
|  | subtype | 数据源子类型 | 字符串 | direct、receiver-based | direct | 数据源子类型，比如kafka数据源可以配置为为direct、receiver-based，其中direct表示以createDirectStream方式创建流，receiver-based表示以createStream方式创建流 | V6.15.60.03B3 |
|  | windowduration | 窗口大小，单位秒 | Int |  | 15 | 窗口大小，如果不进行窗口操作，该字段可以不填 | V6.15.60.03B3 |
|  | slideduration | 窗口滑动大小，单位秒 | Int |  | 5 | 窗口滑动大小，如果不进行窗口操作，该字段可以不填 | V6.15.60.03B3 |
|  | topics | Kafka数据源topic | 字符串 |  | kafka-spark-demo | Kafka数据源topic，当连接多个topic时，以逗号分隔 | V6.15.60.03B3 |
|  | groupid | Kafka 数据源groupid | 字符串 |  | zte-bigdata-realtime-sync | 不存在时，可以不填 | V6.15.60.03B3 |
|  | decoder name | 数据源解码类名称 | 字符串 |  | stringDecoder | 用于找对应的rdd 算法 | V6.15.60.03B3 |
|  | type | 数据源解码类型 | 枚举 | decoder | decoder |  | V6.15.60.03B3 |
| cachetable | name | 需要cache的外部表 | 字符串 |  |  | 需要cache的外部表，理论上流中用到的外部表都应该先cache，以提高运行效率，但数据会更新的表不能进行cache | V6.15.60.03B3 |
|  | type | Cache表类型，缺省为spark，目前也只支持spark | 字符串 | spark | spark | Cache表类型，缺省为spark，目前也只支持spark | V6.15.60.03B3 |
|  | dbname | Spark表所在的数据库 | 字符串 |  | zxvmax |  | V6.15.60.03B3 |
|  | columns | 需要cache的外部表列 | 字符串 |  |  | 多个字段时以逗号分隔 | V6.15.60.03B3 |
| resource | core | 流运行所需要的spark核数占总资源的百分比 | Int |  | 10 |  | V6.15.60.03B3 |
|  | memory | 流运行所需要的spark内存数占总资源的百分比 | Int |  | 10 |  | V6.15.60.03B3 |
| algorithm | name | 流内子任务名称 | 字符串 |  | lte\_qcell\_userinfo | 必须与算法xml中algorithm name一致 | V6.15.60.03B3 |
|  | type | 流内子任务类型 | 枚举 | decoder、transform、output | transform | 流内子任务类型：decoder、transform、output，必须与算法xml中算法类型一致；  Decoder： | V6.15.60.03B3 |

#### 示例说明

算法XML文件说明

算法为流处理过程中的各个子过程的定义，主要包括：

1）算法名称、算法类型、jar包位置；算法类型分为三类：

decoder：流解码，其输入是接入的topic，输出为DStream

transform: 流转换，其输入是DStream,输出为DStream

output：流输出，其输入是DStream,输出为外部表或者文件

2）算法输入名称、RDD类型；

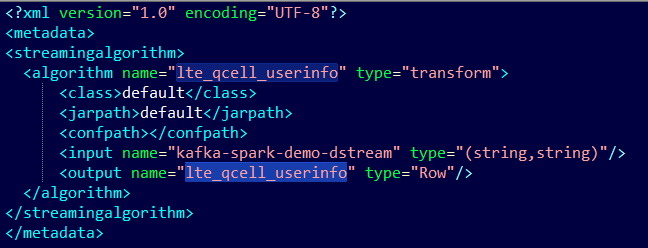
3）算法输出名称、RDD类型

#### 字段说明

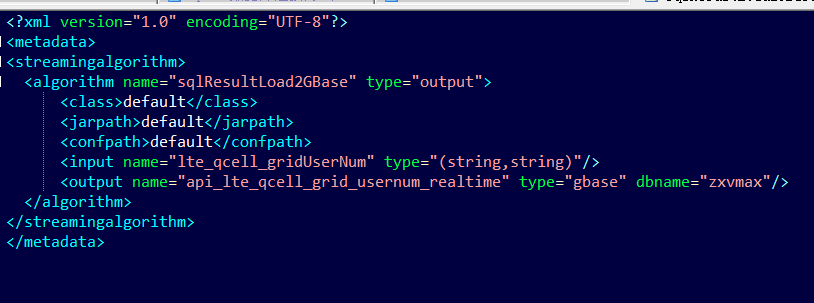
| **字段名** | **子字段** | **字段说明** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| algorithm | name | 算法名称 | 字符串 |  | lte\_qcell\_userinfo | 算法名称，需要与task xml中的algorithm name对应 | V6.15.60.03B3 |
|  | type | 算法类型 | 枚举 | decoder、transform、output | transform | 算法类型，需要与task xml中的algorithm type对应  Decoder：算法的输入是流数据源  Transform：算法的输入和输出  Output：算法目的是输出外部表 | V6.15.60.03B3 |
|  | class | 算法入口class完整名称 | 字符串 |  | com.zte.bigdata.alg | 算法入口class完整名称：例如com.zte.bigdata.alg | V6.15.60.03B3 |
|  | jarpath | 算法jar包位置 | 字符串 |  | %algorithmpath%/lte/feature/realTimeLocation.jar | 算法jar包位置：例如  %algorithmpath%/lte/feature/realTimeLocation.jar | V6.15.60.03B3 |
|  | confpath | 算法配置文件路径 | 字符串 |  | %algorithmpath%/lte/feature/realTimeLocation.conf | 算法在使用自定义配置时，通过com.zte.vmax.metadata.SparkStreaming.SparkStreamingConf.config或者getStreamingEnv. taskConfig获取，例如：SparkStreamingConf.config.getInt("xxx.xxx")；无配置时该项为空 | V6.15.60.03B3 |
|  | input name | 算法输入名称 | 字符串 |  | kafka-spark-demo-dstream | 相当于离线任务的输入表名 | V6.15.60.03B3 |
|  | type | 算法输入DStream[T]中的T的类型 | 字符串 |  | (string,string) | 算法输入DStream[T]中的T的类型，例如从kafka接入的数据key类型是string,值的类型也是string，则其类型为(string,string) | V6.15.60.03B3 |
|  | output name | 算法输出名称 | 字符串 |  | lte\_qcell\_userinfo |  | V6.15.60.03B3 |
|  | type | 算法输出类型 | 字符串 |  | Row | 算法输出类型，当算法类型为decoder或者transform时，其输出为DStream[T]，则这里的输出类型为T；当算法类型为output时，其输出为外部表，这里的类型为：spark、gbase分别表示spark或gbase表 | V6.15.60.03B3 |
|  | dbname | 输出的外部表所在的库名 | 字符串 |  | Zxvmax | 当输出为外部表时，输出的外部表所在的库名，当不输出外部表时，该字段不填 | V6.15.60.03B3 |

#### 示例说明

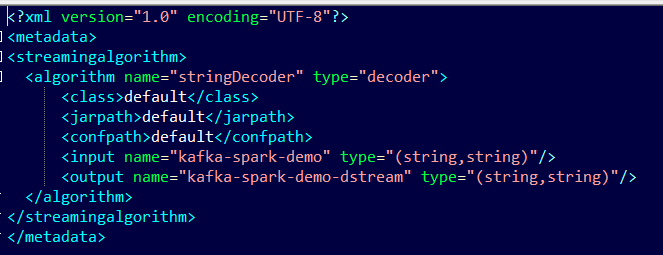
示例一：type为transform的示例



示例二：type为output的示例



示例三：type为decoder的示例



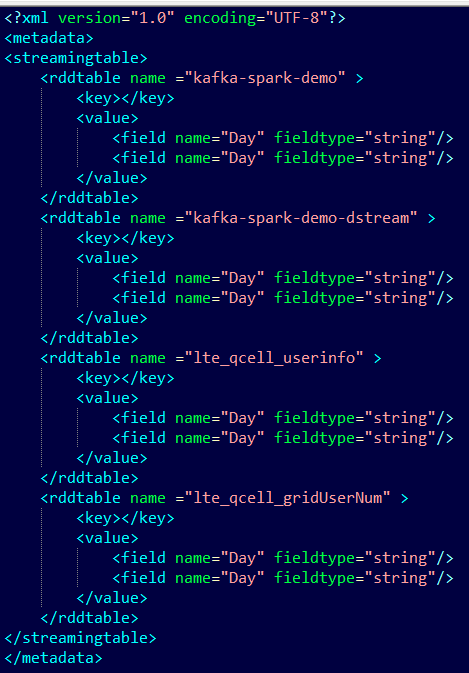
### 表XML文件说明

输入、输出过程RDD每行各个字段的定义，包含字段名称和类型；用于各个算法的对接以及创建临时表。

#### 字段说明

| **字段名** | **子字段** | **字段说明** | **类型** | **取值范围** | **示例** | **说明** | **平台版本** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| rddtable | name | 相当于离线任务的表名 | 字符串 |  | kafka-spark-demo | 相当于离线任务的表名 | V6.15.60.03B3 |
|  | key | Key字段名称 |  |  |  |  | V6.15.60.03B3 |
|  | type | Key字段类型 |  |  |  |  | V6.15.60.03B3 |
|  | value field name | value字段名称 |  |  | Day |  | V6.15.60.03B3 |
|  | value field type | value字段类型 |  |  | String |  | V6.15.60.03B3 |

#### 示例说明



## RDD算法jar包实现说明

Streaming流任务算法全部以RDD jar包方式实现，包括四种算法类型的jar包：decoder、transform、output以及辅助操作算法jar包，以下分别进行说明

### 算法类型说明

#### decoder算法类型实现说明

适用条件：只有kafka direct类型的stream需要提供

decoder算法继承于com.zte.vmax.metadata. CreateStreamBase[K,V,KD <: Decoder[K]:,VD <: Decoder[V]:]，其中K为解码后输出的key的类型，V为解码后输出的value的类型，KD为key解码类类型，VD为value界面类类型；decoder算法继承于com.zte.vmax.metadata. CreateStreamBase只是为了提供K，V，KD，VD类型，不需要重写任何方法（注意：子类必须是class，不能是object）：例如：

class CreateStringStream extends CreateStreamBase[String,String,StringDecoder,StringDecoder]

VK，和VD也需要在decoder jar包中定义，并且继承于kafka.serializer.Decoder实现其fromBytes(bytes: Array[Byte])方法，例如：class StringDecoder(props: VerifiableProperties = null) extends Decoder[String]；

特别说明：

**1、对于输入输出的key和value都是string类型，而且不需要进行字段筛选的decoder，是不需要定义解码jar包的，当不定义jar包时，算法xml中class和jarpath字段填空；**

**2、当一个task需要接入多个流时，需要定义多个decoder算法，其算法名称algorithm name、输出名称output name不能相同;**

#### transform算法类型实现说明

transform算法继承于com.zte.vmax.metadata.StreamingTransformBase[I,O]，重写其transformDStream(in: List[DStream[I]]): List[DStream[O]]方法，其中I为输入DStream的类型，O为输出DStream的类型（注意：子类必须是class，不能是object），例如：

class xxxTransform extends StreamingTransformBase[String,String]

特别说明：

**1、当一个算法有多个输出时，其transformDStream函数的输出顺序必须与算法xml中配置的输出顺序一致**

#### output算法类型实现说明

output算法继承于com.zte.vmax.metadata.StreamingOutputBase[I]，重写其outputDStream(in: List[DStream[I]]): Unit方法，其中I为输入DStream的类型（注意：子类必须是class，不能是object），例如：class xxxOutput extends StreamingOutputBase[String]

#### 辅助操作算法类型实现说明

辅助操作算法继承于com.zte.vmax.metadata.StreamingAuxOperation，重写其beforeStreamCreate()和afterStreamStop()方法，可以只重写其中的一个；（注意：子类必须是class，不能是object），例如：class xxxOperation extends StreamingAuxOperation

### 接口说明

#### 参数信息获取接口

**transform、output、辅助操作 类型算法中ssc、hc、gbase、taskinfo、taskConfig信息的获取：**

ssc: getStreamingEnv.ssc 或者getStreamingEnv. sparkStreamingContext

hc: getStreamingEnv.hc 或者getStreamingEnv.hiveContext

gbase: getStreamingEnv. gbaseInfo 或者getStreamingEnv. gbase

log: getStreamingEnv. logger 用来写日志

taskConfig: getStreamingEnv. taskConfig 任务/算法自定义配置

taskInfo: getStreamingEnv. taskInfo

目前taskinfo中包含了**数据源的窗口大小和每次滑动大小**，以数据源的名称作为索引。例如流任务xml中配置了两个数据源，datasource name分别配置为"source1"、"source2"，其(windowduration, slideduration)分别配置为：(10,5) (15,10)，则taskInfo中的dataSourceInfo信息为：List(("source1",10,5), ("source2",15,10))

#### 输入输出表信息获取接口

**transform、output类型算法中 输入输出表信息的获取：**

**调用getAlgInfo，返回xml中配置的算法信息：包括算法名、类型、输入、输出**

**AlgInfoForApp(algName:String,algType:String,input:List[TableInfo], output:List[TableInfo])**

**其中输入输出信息包括：名称（类型于表名）、类型、数据库名称、表字段（字段名，类型）**

**TableInfo(name: String, rddType: String, dbName: String, streamingRddTable: StreamingRddTable)**

**StreamingRddTable(tableName: String, key: List[RddField], value: List[RddField])**

**RddField(name: String, fieldType: String)**

# 注意事项

## **任务**设计标准

1. 一个清洗任务只输出一张表，只支持spark上清洗，不支持sybase/hbase清洗
2. spark->sybase/hbase/webgis拆成两个清洗任务

1）spark->spark

2）spark生成sybase/webgis/hbase文件，这个任务无算法实体（实体由元数据完成），

只需写算法xml

1. 临时表：只被一个清洗任务使用则放到清洗任务内，被多个清洗任务使用则提成正式表有单独的清洗任务生成这张表
2. 定时执行任务（非数据驱动任务），

如sybase->sybase任务，需采用定时周期任务，算法自己打jar包，算法xml填写注意

<algorithminfo subtype="sybasetimedrivejar">，示例如下：



## 开发注意事项

**1、spark同步到sybase流程**：

1）spark->文件->sybase，其中生成文件和文件同步到sybase过程由平台定时统一完成，业务需要识别是哪张spark表要同步到sybase即可。

案例：spark同步到sybase过程中，又做了一些关联的处理，如spark生成表a，表a和b关联直接生成文件，文件同步到sybase

修改方式：表a和b生成个表c，但是表c的名字不能影响sybase表名，即不影响web查询。

2）由于导出成sybase文件后，sybase入库之前需要删除重复数据。如果sybase表的时间字段不是标准的，需要在sybase汇总表中填写时间的字段信息，目前利用汇总表中的两个属性partitionfield和partitionfieldtype来填写。例如，如果是周和月的表有多个时间信息，则用逗号分隔：

<field fieldname="terminalanalysis\_voice\_qoe\_day" tablename="terminalanalysis\_voice\_qoe\_day" tabletype="sybase" classify="union" source="vmax" timegranularity="1day" version="v1.0" description="" createtime="2015-12-09" storedtype="" dbname="zxvmax" location="" partitionfield="day" partitionfieldtype="timestamp" isexternal="0"/>

**2、spark同步到webgis流程**（专指有预处理文件的形式，web查询触发直接生成的webgis文件不在此处理）：spark->文件->webgis，其中生成文件和文件同步到webgis过程平台统一定时完成，但是业务需要识别出是哪张spark表要同步到webgis。

案例：spark同步到webgis过程中，又做了一些关联的处理，如spark生成表a，表a和b关联直接生成文件，文件同步到webgis

修改方式：表a和b生成个表c，但是表c的名字不能影响webgis文件名，即不影响web查询。

**3、cache问题**：梳理etl时，识别出那些清洗有cache操作即可（要在excel表中标识出），缺省配置为不cache。

**4、时间分区规范化**

1）周分区，目前如下两种方式都可以，如下：  
 年+周，int+int

年+周+天，int+int+string

2）月分区，目前如下两种方式都可以，如下：

年+月，int+int

年+月+天，int+int+string

3）天分区：天，string

小时分区：天+小时，string+int

4）分区命名根据建表实际配置的分区名来填写

5）输入表支持的分区组合为：

1. p\_date

每天生成一个公共牌上的一个灯；

1. p\_imsi、p\_date

每天每一个imsi分区生成一个公告牌上的灯，imsi的分区个数由配置文件配置，目前缺省配置个数为11；

1. p\_date、p\_hour

每天每小时分区生成一个公告牌上的灯；

1. p\_year、p\_month

每月生成一个公告牌上的灯，该灯放在哪天的公告牌上，由配置文件配置，缺省为下月的第一天；

1. p\_year、p\_week

每周生成一个公告牌上的灯，该灯放在哪天的公告牌上，由配置文件配置，缺省为下周的第一天；

1. p\_year、p\_month、p\_date

每天生成一个公告牌上的一个灯；

1. p\_year、p\_week、p\_date

每天生成一个公告牌上的一个灯；

1. p\_year、p\_week、p\_hour

每天每小时生成一个公告牌上的一个灯；

6）无线侧，所有表建表脚本统一使用分区名如下：

p\_provincecode、p\_imsi、p\_year、p\_month、p\_week、p\_date、p\_hour

7）核心网，所有表建表脚本统一使用分区名待定

**5、周任务**需要改为昨天周+今天天或每周执行一次，不支持周任务每天执行并且执行前几天的情况

**6、算法xml执行周期使用如下名字**：”provice，year”、 ” provice，month”、 ” provice，week”、 ” provice，day”、” provice，hour”。

# 特殊功能配置说明

## **Sybase\gbase同步任务的配置修改**

在 dataarrivalcheck.conf 有配置 spark与sybase 表的分区对应关系表，所有现有表的分区字段都要加入到里面，否则会出现spark表同步sybase表失败的情况。

**修改方式：**目前配置如下，业务功能如有新的同步字段类型

*PartitionRelation{*

*p\_hour = ["hour"]*

*p\_date = ["day","datestamp","endtime","timestamp,"reportdate"] // 在 p\_date 字段中 需要 添加 "reportdate"*

*p\_week = ["week"]*

*p\_month = ["month"]*

*p\_year = ["year"]*

*p\_provinceid = ["province","provinceid"]*

*p\_provincecode = ["provincecode"]*

*p\_imsi = ["imsi"]*

*}*

**生效方式：**修改该配置后，重启工程即可生效

## **忙闲时文件配置**

**目的：**定义忙时、闲时时间段，用于业务进行忙闲时数据处理

**修改方式：**修改方法分为两步，第一步定义忙闲时时段，第二步指定表的忙闲时配置，详细说明如下：

1）定义忙闲时时间：在etl工程config/timesetting.conf目录下配置timesetting.conf文件；忙闲时时间分为两类，一类为全系统策略使用timesetting参数，即timesetting中的时间段有效范围为整个系统，其中的busymorningtime，busynighttime，idletime业务可自定义名称，时间段设置必须为连续的，不能为离散值，且小时与小时之间必须以斜杠分开；另一类为用户自定义时间段，定义单个表的忙闲时时段，格式为【表名=忙闲时时段】。tablename = 11/12/13，是单个表的忙闲时设置，不区分忙闲时，表明该表任务只在指定时段运行，其他时段不运行，供后续用户修改单个表的忙闲时，优先级比xml中的高如下：

timesetting{

busymorningtime=4/5/6

busynighttime=8/9/10

idletime=11/12/13

}

//用户单个表的忙闲时修改

usertabletimesetting{

tablename = 11/12/13

}

2）使用方法:算法xml中的忙闲时字段配置见[算法XML文件说明](#_算法XML文件说明)中的timesetting字段说明 

**生效方式：**修改该配置后，重启工程即可生效

## **数据到达检查相关配置**

在汇总表中source字段配置为netmax、dcu的表将进行到数据到达检查。以下说明数据到达检查的规则。

1. 汇总表source字段

source字段，填写原则：

1. cn侧的数据源表配置为 dcu，cn侧清洗出的表配置为vmax；
2. ran侧需要到达检查的netmax数据表则配置为netmax ，中间清洗数据表需要到达检查配置为vmax；
3. 工参、字典表、定时任务等没有输出若需要到达检查则配置为netmax
4. 汇总表datasourcetype / isDynamic、tabletype字段

数据到达检查旧方式：检查类型是根据 isDynamic和tabletype的组合类型判断的；

数据到达检查方式进行了建模优化，建模之后，新增一个字段datasourcetype，该字段决定了检查类型，同时对之前的配置也做了兼容。后续新增表需按照新方式配置到达检查方式。

新旧配置的对应关系如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **source** | **datasourcetype** | **tabletype（新的检查方式）** | **isDynamic** | **Tabletype（旧的）** | **检查规则** |
| dcu | dcu | **netmax** | 任意 | 任意 | Cn自行实现检查 |
| netmax | spark\_partition | **netmax** | 任意 | spark | 检查表分区 |
| netmax | spark\_txt\_file | **netmax** | true | spark\_txt\_file | 检查文件内容 |
| netmax | spark\_xml\_file | **netmax** | true | spark\_xml\_file | 检查文件个数 |
| netmax | spark\_ems\_file | **netmax** | true | spark\_ems\_file | 不检查 |
| netmax | noDynamic | **netmax** | false | 任意 | 检查文件是否存在 |
| netmax | gbase\_import | **netmax** | 任意 | 任意 | 检查gbase中是否有数据 |
| netmax | broadband | **netmax** |  |  | 检查文件目录，不含数据中心分区(目录结构类似/xx/yy/zz/p\_provincecode=5100/p\_date=2016-06-06,判断准则为某省当天目录下有文件且该省存在之后时间目录，如p\_provincecode=5100若想翻2016-06-06号的牌，必须满足该天目录下有文件，而且2016-06-06之后的目录也存在，比如2016-06-07、2016-06-08等) |
| netmax | iptv | **netmax** |  |  | 检查文件目录，含数据中心分区(目录结构类似 /xx/yy/zz/p\_provincecode=5100/p\_datacenter=22/p\_date=2016-06-06判断准则为某省的某数据中心当天目录下有文件且该数据中心存在之后时间目录，如p\_provincecode=5100、p\_datacenter=22数据中心若想翻2016-06-06号的牌，必须满足该天目录下有文件，而且2016-06-06之后的目录也存在，比如2016-06-07、2016-06-08等) |
| netmax | streaming\_partition | **netmax** |  |  | 检查表分区，当前分区存在且该时间点之后的某分区也存在 |
| netmax | spark\_txt\_file2 | **netmax** |  |  | 检查文件内容，如果为空，则检查文件名 |

说明：

1. datasourcetype的配置依据业务对表的检查需求，按照检查规则确定；
2. 在汇总表中，如果配置了datasourcetype字段，那么就以此字段为准，如果没有配置，则按照旧的检查类型进行判断，这样能够兼容之前的配置
3. 由于gbase\_import检查类型是新的检查方式，因此只能通过配置datasourcetype字段生效，老的检查类型是没有对应于这种检查方式的
4. 建议新增的表配置datasourcetype字段来决定检查方式

对于新增的数据源表，每种检查方式所需的配置说明如下（针对新的配置方式，已有表的配置方式不需修改）：

|  |  |
| --- | --- |
| 检查方式 | 配置项（汇总表中） |
| dcu | **source**：配置为dcu  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为dcu |
| spark\_partition | **source**：配置为netmax  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **tabletype:** 配置为netmax，该字段已经不作为判别检查方式的依据，但是为了和spark类型区分，统一设置为netmax  **datasourcetype**：配置为spark\_partition |
| spark\_txt\_file | **source**：配置为netmax  **tabletype:** 配置为netmax，该字段已经不作为判别检查方式的依据，但是为了和spark类型区分，统一设置为netmax  **location**: drs推送至hdfs上的目录  **timeindex**：文件中时间字段所在列  **timeformat**：时间format的格式，默认为“yyyy-MM-dd HH:mm:ss”，如果填写为“timestamp”，则表示从2000-01-01到目前时间的秒数  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为spark\_txt\_file |
| spark\_xml\_file | **source**：配置为netmax  **tabletype:** 配置为netmax，该字段已经不作为判别检查方式的依据，但是为了和spark类型区分，统一设置为netmax  **location**: drs推送至hdfs上的目录  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为spark\_xml\_file |
| spark\_ems\_file | **source**：配置为netmax  **tabletype:** 配置为netmax，该字段已经不作为判别检查方式的依据，但是为了和spark类型区分，统一设置为netmax  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为spark\_ems\_file |
| noDynamic | **source**：配置为netmax  **tabletype:** 配置为netmax，该字段已经不作为判别检查方式的依据，但是为了和spark类型区分，统一设置为netmax  **location**: drs推送至hdfs上的目录  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为noDynamic |
| gbase\_import | **source**：配置为netmax  **tabletype:** 配置为netmax，该字段已经不作为判别检查方式的依据，但是为了和spark类型区分，统一设置为netmax  **partitionfield**：gbase表的分区字段名（必填）  **partitionfieldtype**：gbase表的分区字段类型  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为gbase\_import  说明：partitionfield字段必须是gbase表中存在的字段，目前分区只包含省分区及时间分区，填写规则可参照gbase同步中gbase表分区配置 |
| broadband | **source**：配置为netmax  **tabletype:** netmax  **location**: drs推送至hdfs上的目录  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为broadband |
| iptv | **source**：配置为netmax  **tabletype:** netmax  **location**: drs推送至hdfs上的目录  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为iptv |
| streaming\_partition | **source**：配置为netmax  **tabletype:** netmax  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为streaming\_partition |
| spark\_txt\_file2 | **source**：配置为netmax  **tabletype:** netmax  **location**: drs推送至hdfs上的目录  **timeindex**：文件中时间字段所在列  **timeformat**：时间format的格式，默认为“yyyy-MM-dd HH:mm:ss”，如果填写为“timestamp”，则表示从2000-01-01到目前时间的秒数  **delaytime**：检查文件到达后，延迟翻牌的时间，时间粒度为分钟  **datasourcetype**：配置为spark\_txt\_file2 |

注：上述配置字段大多数具有默认值，默认值配置在metadata-datamanage.conf文件中的summarytabledefaultvalue配置项中

1. 配置文件修改

1)datasourceconfig.xml文件：设置netmaxid,用于datastorage入库任务的数据到达检查

2)详见《元数据平台环境配置说明》章节2.4

## **数据生命周期配置**

**目的**：系统运行一段时间后，会产生大量的数据；为了节约存储空间，提高系统运行效率等，需要对数据制定一定的数据存放和删除等管理策略。目前的数据生命周期管理策略用于控制数据的存放时长，按数据源类型分为不同数据存放策略；每个表需配置一个数据存放策略。存储时间到达时，即数据的生命周期结束，需对数据进行相应的删除操作。

数据生命周期相关控制参数说明如下：

1. 数据生命周期功能开关

数据生命周期开关配置在./vmax-conf/dataperiodmanage.conf文件中，默认开关不打开；不打开时生命周期相关的配置不生效，在开关打开时，数据生命周期的配置才生效。

UserPeriodSwitchState{

state = false # 用户配置该功能是否可用，默认是不可用false

}

1. 通用数据生命周期的策略类型

在dataperiodmanage.conf中配置DataStoredPeriod类型的保存时间周期，配置500就是按当天时间保存500天数据；DataStoredPeriod中的类型名称用户可自定义，该参数可供整个系统使用。

示例：# 数据生命周期管理模板，GbaseDefaultType是gbase默认保留天数;

# SparkDefaultType是spark默认保留天数;

# SparkNetmaxType是spark的netmax表保留天数.

DataStoredPeriod{

GbaseDefaultType = 500

SparkDefaultType = 500

SparkNetmaxType = 500

}

1. 用户自定义数据生命周期策略

用户如果要设定特定的表的保留时长，就在CustomStoredTablePeriod配置项中增加表名和对应的保留天数；如果要设定spark某个hdfs路径下的所有表的保留时长，则添加hdfs路径和对应的保留天数。特定表设置的优先级最高，其次是hdfs路径的配置，最后是通用的策略类型。

示例：

# 用户特定表或者hdfs路径(需要是spark表对应的hdfs路径，而且要在汇总表里面有定义)保留天数

CustomStoredTablePeriod{

ExampleTableName = 500

/Example/Hdfs/Path/ = 500

}

1. 表的数据生命周期配置方式

业务要在汇总表summarytable中配置表的数据生命周期方式，需要将storedperiod字段配置为DataStoredPeriod中的其中一种类型或CustomStoredTablePeriod中的特定表名，如SparkDefaultType（netmax表对应SparkNetmaxType），表示数据保存500天。示例：

<field fieldname="ict\_subject\_terminalanalysis\_data\_qoe\_month"

tablename="ict\_subject\_terminalanalysis\_data\_qoe\_month" tabletype="spark" classify="union"

source="vmax" timegranularity="1day" version="v1.0" description="" createtime="2015-12-09"

storedtype="textfile" dbname="zxvmax"

location="/zxvmax/telecom/ict/subject/ict\_subject\_terminalanalysis\_data\_qoe\_month/"

partitionfield="p\_provincecode,p\_year,p\_month,p\_date" partitionfieldtype="int,int,int,string"

isexternal="1" storedperiod="SparkDefaultType"

virtualpartitionvalue=""/>

1. 数据删除

平台根据汇总表xml中的storedperiod字段，确定每个表的数据生命周期，在配置的数据生命周期结束时，系统进行数据删除。数据删除操作的对象表包括spark、sybase、gbase等，由于系统需要利用表中数据的时间字段为依据进行数据删除操作，因此，需要准确确认各表的时间字段。数据删除的策略和时间，由业务维护和配置，如果没有配置删除策略，缺省不删除。对于每种数据库，元数据平台默认支持通用的时间字段，对于特殊的时间字段，需要额外进行配置，配置要求如下：

1. 用户特定字段—sybase表时间配置

sybase表的时间字段设置默认为day、week 、month和year，如果在表中字段找不到就在UserSybaseTimeField里面找，业务可以后期维护。

sybase示例：# 用户特定字段配置 sybase配置特殊时间字段 优先级最高为day

UserSybaseTimeField{

day = day

year = year

month = month

week = week

datatime = day

}

1. 用户特定字段—gbase表时间配置

gbase表的时间字段设置,如果在表中字段找不到就在UserGbaseTimeField里面找，业务可以后期维护；Gbase示例：# 用户特定字段配置 Gbase配置特殊时间字段 优先级最高为day，把标准的字段放在后面

UserGbaseTimeField{

hour = day

datatime = day

datestamp = day

date = day

reportdate = day

day = day

year = year

month = month

week = week

}

1. 用户特定字段—spark表时间配置

spark表的p\_date分区一定要符合规范的 年-月-日 这样的格式的，对于不符合格式的删除会失效，spark表的时间字段配置；示例：# 用户特定字段配置 spark配置天分区字段，把标准的字段放在后面

UserSparkPartitionField{

reportdate = p\_date

p\_date = p\_date

p\_year = p\_year

p\_month = p\_month

p\_week = p\_week

}

1. 核心网有自己的数据维护策略，核心网的都不填storedperiod属性（summaries\_tab\_cn\_fact.xml、summaries\_tab\_cn\_aggr.xml）
2. 数据生命周期删除任务的运行周期

数据生命周期删除任务执行的周期决定了数据删除的频率，即数据生命周期删除任务一个月运行一次，则每月删除一次数据，一周运行一次，则每周删除一次数据，业务可根据数据量大小配置删除的频率。若数据或spark上表分区过多（例如，四川），如一个月删除一次，可能导致spark数据分区数删除不干净；需要更改任务的执行策略为一周删除一次，改完后重启元数据进程生效。

配置方法为修改spark\_data\_period.xml和gbase\_data\_period.xml中的任务执行周期正则表达式，执行周期配置举例：

每月执行：cron= "0 0 0 1 \* ?"：每一个月1号凌晨0点0分执行一次

每周执行：cron="0 40 0 1,7,14,21,28 \* ?"：每个月1,7,14,21,28号的凌晨0点40分执行

xml任务配置文件位置是在安装vmax后的ICT服务器上，具体文件位置：

/home/netnumen/ems/ums-server/utils/vmax-metadata-manager/metadata/dataperiodmanage/feature/dataperiod/algorithm/spark\_data\_period.xml

/home/netnumen/ems/ums-server/utils/vmax-metadata-manager/metadata/dataperiodmanage/feature/dataperiod/algorithm/gbase\_data\_period.xml

1. 任务所在的路径信息配置，不能修改

DataPeriodFilePath = {

path = "./dataperiodmanage/feature/dataperiod/algorithm"

}

## **实时任务的资源配置**

实时任务是指，任务的执行条件满足时（数据到达或者定时的时间到达），需要立即执行。在原有系统中需要在就绪队列中等待资源，不满足实时性。因此实现了资源预留的功能，来满足实时执行的需求。但是预留有个缺点，预留的资源有空闲的可能，造成浪费资源

1、元数据平台预留资源的配置方法

修改vmax-conf/datanotifyboard.conf，如下红色标注的即为实时资源池的配置，把#号去掉重启平台，就可以生效。具体各自字段含义见下，也可以根据需要调整资源池各字段的值，也可以新建资源池。

vmax-conf/datanotifyboard.conf

//任务公告牌配置

taskboard{

totalCore = 60

totalMem = 180

availableCore = 60

availableMem = 180

checkdatarealtime = 0

percentOfCacheCore = 10

percentOfCacheMem = 10

pool = [

# "name=cachePool;priorityRange=[100,199];percentOfCore=10;percentOfMem=10;lendEnable=true",

# "name=realTimePool;priorityRange=[200,299];percentOfCore=10;percentOfMem=10;lendEnable=false"

# 资源池配置说明

# name 资源池的名字,用于索引资源池,名字不能相同

# priorityRange 优先级下界和上届，资源池是通过优先级表示任务的归属关系，例如任务优先级为100，资源池priorityRange=[100，199]，则任务属于这个资源池

# percentOfCore 该资源池的核数占总资源核数的比例，建议10的倍数，避免出现资源碎片

# percentOfMem 该资源池的内存占总资源内存的比例

# lendEnable 表示是否可以借给其他池的任务使用

# 默认有一个commomPool 资源为总资源减去其他各资源池的和 lendEnable=True priorityRange=[0，0]

# 资源竞争规则

# 同一个资源池的任务，先按照优先级高低排序，再按照到达顺序排序

# 如果一个资源池 lendEnable=FALSE，不属于该池的任务不允许借资源

# 如果一个资源池 lendEnable=TRUE，如果不属于该池任务，并且优先级高于该池的优先级下限，则允许该任务借资源

# 如果一个资源池 lendEnable=TRUE，如果有任务属于该池，则优先分配资源，即使存在非本池的更高优先级的任务

]

}

生效方式：把#号去掉重启平台，修改该配置后，重启工程即可生效

2、实时任务需要配置优先级，使任务的优先级处于datanotifyboard.conf中对应的资源池的范围内；任务的优先级是在虚拟任务的XML中配置的：

任务优先级示例：



priority为优先级 配置为201，即属于realTimePool资源池中优先级的配置。

目前任务优先级只能在虚拟任务级别配置，因此，实时任务需单独处于一种虚拟任务中。

3、资源比例

realTimePool资源池的资源比例是10%。如果实时任务需要更大的资源，也需要修改资源池中预留的资源比例。但是这存在浪费更多资源的风险。

## **多省市业务配置**

多省市接入业务，相关的配置文件有两个，汇总表xml和vmax-conf/datasourceconfig.xml

**修改方式：**

1）汇总表xml中的classify字段，配置表的依赖数据的实际制式；

2）在vmax-conf/datasourceconfig.xml中配置provincecode参数，其中，dataclassify制式需要汇总表xml中的制式classify一致，否则会导致数据公告牌不能正确翻牌，具体配置说明如下：



## **数据公告牌可视化分类配置**

目的：Web页面的数据公告牌，用户可分类定义视图进行监测。

修改方式：需要在…/vmax-conf /databoardview.conf中修改views和对应的视图表内容。缺省视图显示配置为：views = [netmax,dcu,api,dk]，其中netmax,dcu是无线和核心网入库表类型，名称不可以修改，表不可配置； api和dk是自定义视图名称。自定义视图中需要监测的表数据需进行配置，如：

api = [ict\_subject\_gridbasicproperty,

ict\_subject\_griduser\_hour,

ict\_subject\_griduser\_day]

dk = [abnormal\_pl,

ceiqoe\_ci,

ceiqoe\_ci\_lte]

**其中，视图下需要监控的表名称，如果不是spark表 需要配置成 表名|表类型 例如 tablexxx|spark\_ems\_file；只配置表名时，默认为spark表。**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **参数小类** | **参数名（英文）** | **参数含义（中文详细作用）** | **取值范围** | **默认值** | **当前生效模式** | **引入平台版本** | **参数类别** |
| 数据公告牌可视化 | databoardview.views | 数据公告牌可视化需要呈现的分类表；在四川局配置为netmax,dcu,api,dk四类，在其他局可去掉api，dk的视图类型，并进行新的自定义 | 字符串 | [netmax,dcu,api,dk] | 重启元数据进程 |  | Ran+cn业务 |
| 数据公告牌可视化 | databoardview.xxx | xxx视图分类下需要呈现的表名，xxx为databoardview.views中的其中之一，其中类别netmax、dcu不用配置表名，自动获取；多个表之间逗号隔开 | 字符串 | api = [ict\_subject\_gridbasicproperty, … cdma\_netmaxc\_nbi\_Forecast\_DO\_Airlink\_Data\_Detail] | 重启元数据进程 |  | Ran+cn业务 |

## 不可到达**netmax**表配置****

目的：当某个netmax节点的某张表数据无法产生时（比如连接异厂商设备时，有些表不会产生），可配置成不可到达表，数据公告牌将其状态置为1，认为数据已经达到。

修改方式：修改…/vmax-conf /datasourceconfig.xml文件，如下：

<dataclassify name="cdma">

<netmaxid>10000,12000</netmaxid>

<virtualpartition></virtualpartition>

**<cannotarrivenetmaxtables>**

**<cannotarrive netmaxid="10000,12000"> <!-- netmaxid -->**

**<tablename>Cannot1,cannot2</tablename> <!-- 该netmaxid 无法到达的数据表-->**

**<tablename>Cannot3</tablename> <!-- 该netmaxid 无法到达的数据表-->**

**</cannotarrive>**

**<cannotarrive netmaxid="10000">**

**<tablename>Cannot4,Cannot5</tablename>**

**</cannotarrive>**

**</cannotarrivenetmaxtables>**

</dataclassify>

## 栅格大小配置说明

目的：webgis同步目前不同制式的栅格大小是固定写死的，如100\*100、50\*50，栅格大小按固定值可能不满足需求，需要提供一个可以根据制式配置栅格大小的接口。

配置方式：

1. 配置整个制式的栅格大小：在../vmax-conf/serviceaddress文件中增加或修改相应制式的栅格大小，格式必须为sld文件名中的制式（小写）+ GridSize的形式，如：

umtsGridSize = 50 //sld文件名中的制式（小写）+ GridSize

1. 配置单个业务的栅格大小：在算法相应的webgis conf配置文件中，针对每个表增加栅格大小的配置值，用gridsize字段表示，如：gridsize = 100

生效优先级：单个业务的表配置了gridsize值，则针对该业务该表优先获取gridsize作为栅格大小生成webgis预处理文件；若单个业务未配置gridsize值，则获取serviceaddres中相应制式的栅格大小；若业务和serviceaddres配置文件都未配置栅格大小，则取默认值100

## 推送表文件到指定ftp服务器说明

目的：

将业务sql算法的计算结果保存为CSV文件，并推送到指定ftp目录下。输出的CSV文件命名规则为：时间+outputtable名.csv。

配置方式：

1）算法xml中，subtype属性配置为fileupload，如<algorithminfo subtype="fileupload">；

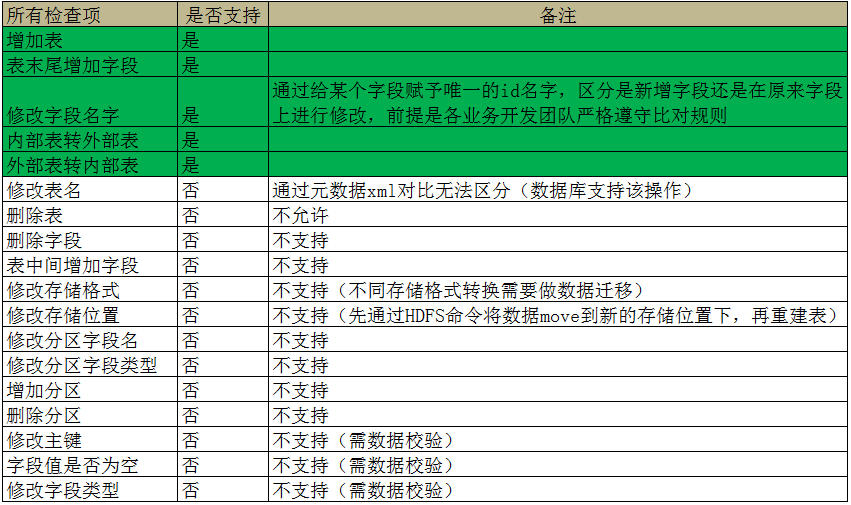
2）在…/ums-server/utils/vmax-metadata-manager/metadata/config/dashboardFtpPath.conf文件中新增ftp服务器相关信息，格式如下；其中ip为ftp服务器地址，port为端口号，user为用户名，pwd为密码，path为ftp上存放的路径；如果为空，需保留双引号。

3）dashboardFtpPath.conf中的配置应用于整个系统，配置为fileupload业务使用同一套ftp服务器配置。



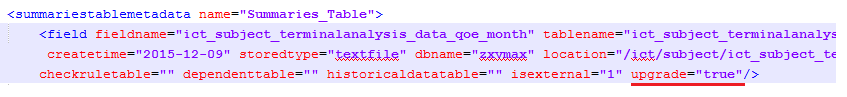
# 表升级规范说明

## **spark**表升级规范

****

1. 汇总表XML 文件中增加"upgrade"字段。"upgrade='true'" 表示表可以升级。"upgrade='false'" 表示表不可以升级。默认不填，表示表可以升级。

示例：



2、表XML 文件中增加"fieldid"字段。修改表字段时，fieldid="旧字段名称"；新增的表字段时， fieldid="新增字段名称"。

<tablemetadata name="ict\_subject\_terminalanalysis\_data\_qoe\_month" tabletype="spark">

<field fieldid="year" fieldname="year" fieldtype="int" notnull="0" length="" isprimarykey="0" description="" algorithm="" source=""/> <--**新增表字段示例** /-->

<field fieldid="month" fieldname="newmonth" fieldtype="int" notnull="0" length="" isprimarykey="0" description="" algorithm="" source=""/> <--**修改表字段示例** /-->

<field fieldid="province" fieldname="province" fieldtype="string" notnull="0" length="" isprimarykey="0" description="" algorithm="" source=""/>

</tablemetadata>

3、用户在修改表字段的时候，需要保证新字段的"名称"、"类型" 准确，元数据不检查数据类型转换的可行性。例如原值为bigint类型，修改后为int类型，是否成功不做检查。

**支持对表修改的操作：**

1）增加字段：表XML在表最后增加一条<field/>。

如<field fieldid="days" fieldname="days" fieldtype="int".../>

2）修改表字段（只能修改名称，不能修改类型）：在表XML中修改对应字段的<field/>属性。

如源表字段：<field fieldid ="days" fieldname="days" fieldtype="int".../>

修改后字段：<field fieldid ="days" fieldname="dayByDay" fieldtype=" int ".../>

3）修改表类型：新生成的汇总表XML中isexternal属性值的修改。

内部表转变为外部表: isexternal="0" -> isexternal="1"

外部表转变为内部表: isexternal="0"

**不支持对表修改的操作：**

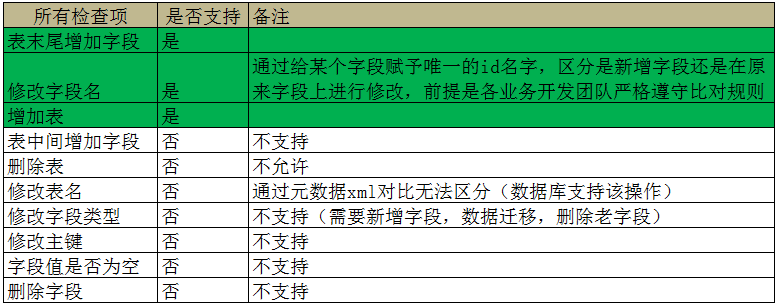
1）对表分区的操作：增加、删除、修改

2）删除 表字段

3）修改数据源 ：location、storedtype

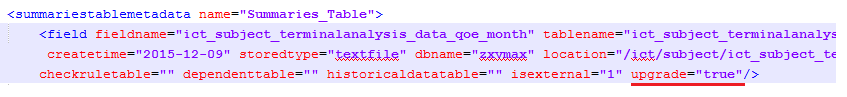
4）不支持删除表

## **Sybase**/Gbase表升级规范



1. 汇总表XML 文件中增加"upgrade"字段。"upgrade='true'" 表示表可以升级。"upgrade='false'" 表示表不可以升级。默认不填，表示表可以升级。

示例：



2、表XML 文件中增加" fieldid "字段。修改表字段时，fieldid ="旧字段名称"；新增表字段时， fieldid ="新增字段名称"。fieldid值不能修改。

<tablemetadata name="ict\_subject\_terminalanalysis\_data\_qoe\_month" tabletype="sybase">

<field fieldid="year" fieldname="year" fieldtype="int" notnull="0" length="" isprimarykey="0" description="" algorithm="" source=""/> <--**新增表字段示例** /-->

<field fieldid="month" fieldname="newmonth" fieldtype="int" notnull="0" length="" isprimarykey="0" description="" algorithm="" source=""/> <--**修改表字段示例** /-->

<field fieldid="province" fieldname="province" fieldtype="string" notnull="0" length="" isprimarykey="0" description="" algorithm="" source=""/>

</tablemetadata>

3、Sybase/Gbase都**不支持字段类型修改**。

**支持对表修改的操作：**

1）在表最后增加字段：新生成的表XML中，增加一条<field/>。

如<field fieldid ="days" fieldname="days" fieldtype="int".../>

2）修改表字段（名称：新生成的表XML中，在fieldid对应的<field/>进行修改。

如源表字段：<field fieldid ="days" fieldname="days" fieldtype="int".../>

修改后字段：<field fieldid ="days" fieldname="dayByDay" fieldtype=" int ".../>

**不支持对表修改的操作：**

1）删除 表字段

2）不支持删除表

3）不支持字段类型的修改