**MO模型库管理设计规范（5G时代）**

# 模型库设计

本文编写的目的是为了各个产品在制作MO模型库包时，按照本文的要求进行编写。

当前网管：网管功能和网元模型逻辑混在一起，升级网元版本时，也要升级网管版本。

扁平化网管：网管功能和网元模型逻辑解耦分离，升级网元版本可以通过导入模型包实现，网管不用升级，对大部分场景，网管进程也无须重启（是否重启网管进程，由各领域对新网元模型包的处理能力决定）。



## 网元模型库管理能力

1. 提供了网元模型包的定义规范。按照该规范实现网元模型和网管功能的解耦。
2. 在网管上实现网元模型包的导入，加载，回退，客户端推送分发。支持热加载模式，也支持按要求重启分进程的需求。

## 模型库文件定义

模型库是一个zip包，通常它包含一个网元版本的和网元业务逻辑相关的内容。模型包有2种形式：

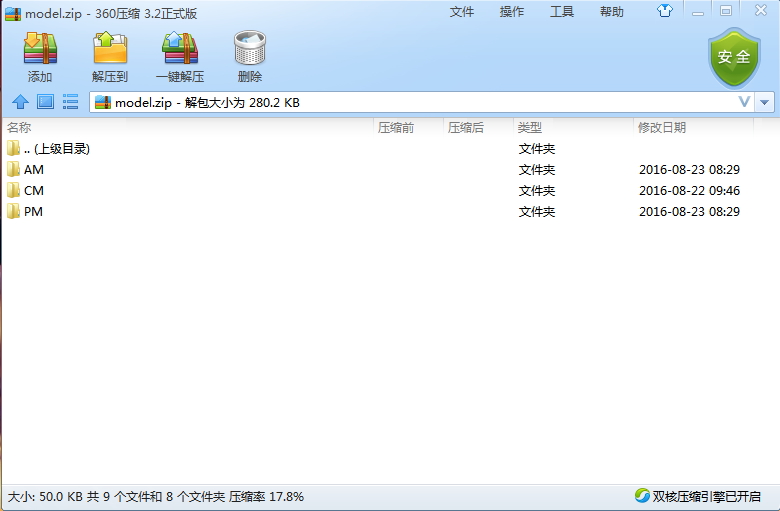
1. 一个网元版本全新的模型包。
2. 一个网元版本的补丁模型包。比如某个网元版本的lua约束有bug，因此出一个模型的补丁包来解决这个问题。还有一种就是针对新增一个物理设备，比如新增rru规格，它也可以算作一种模型补丁包。
3. 网元模型包不光是配置模型，还应该包括告警（告警码），动态，诊断，性能（计数器）等各个领域与网元模型相关的内容。
4. 网元模型包的发布和网元版本是绑定的，它的发布时间和网元版本发布时间保持一致。

配置的模型库有一些特别的地方，如下：

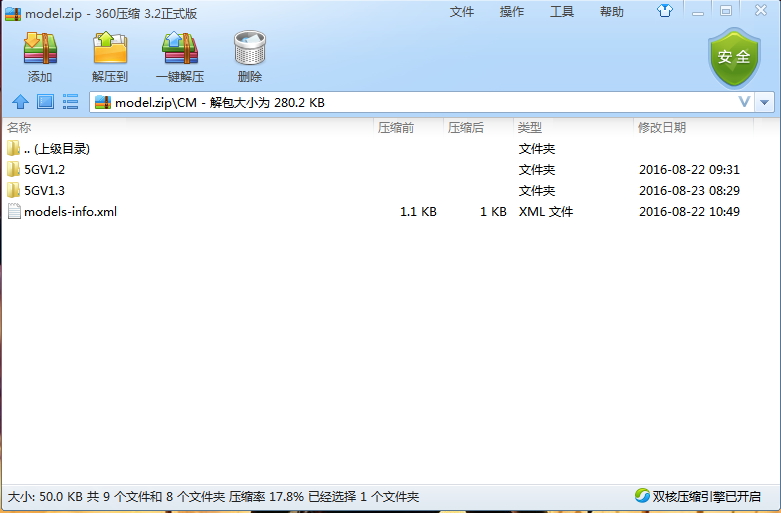
* 不同的网元版本，模型定义都不同。即管理多少个网元版本，就必须有多少个网元版本的模型定义。而其他模块，可能只需要定义一套最新的模型库，向下兼容老网元版本即可。
* 配置模型不光定义网元模型，还定义了网元下的各个制式的模型。各个制式模型可以像搭积木一样组合成网元模型。比如一个层三平台的制式模型，可以用于多个网元版本。

### 模型包文件结构

模型包文件是一个zip文件，通过网管的导入并激活到系统后形成模型库的工作目录。模型包文件先按业务领域划分，比如cm，ddm，fm，然后每个领域内再存放领域内部的模型文件。下面是模型包压缩文件内部第一层的目录示范：

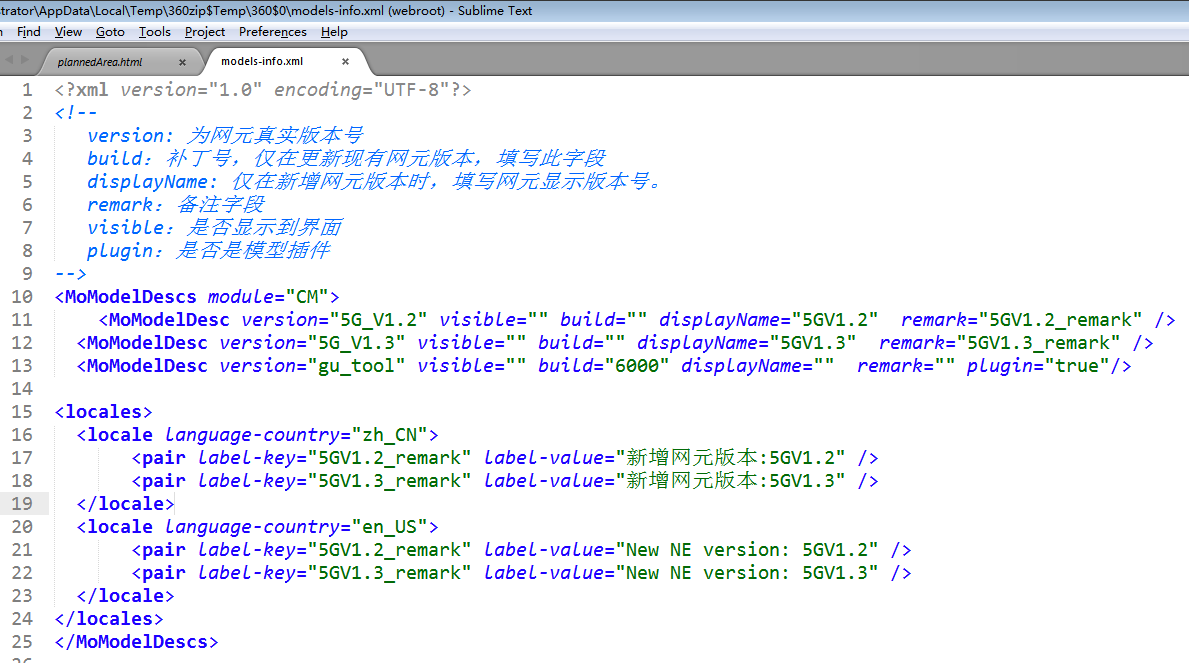


下面以配置领域的模型包结构为例（其他模块类似）如下图所示：



每个领域的模型包中必须有一个models-info.xml文件描述模型信息，models-info.xml文件如下图所示，它体现了网元模型类型及相关网元版本信息：

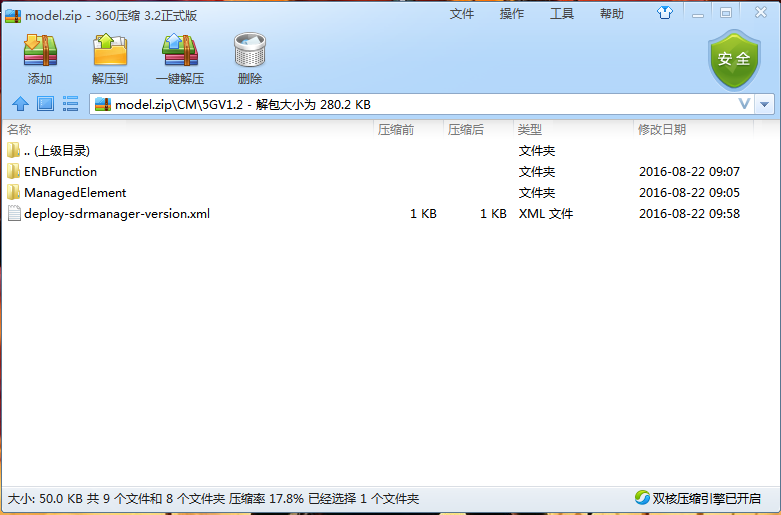
新增版本示例：



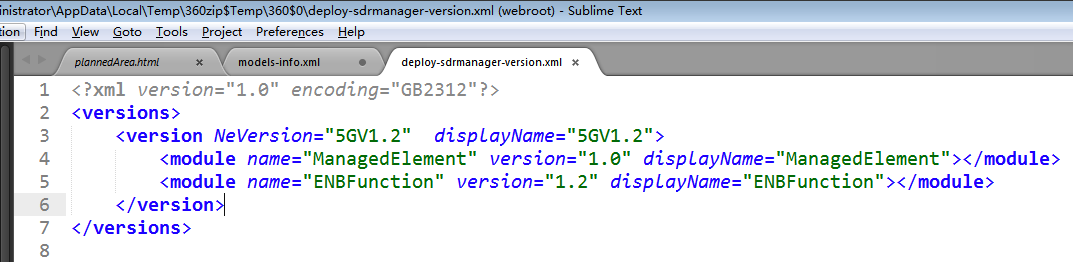
* module: 模块名称，如配置CM,动态DDM。模型库系统按此名称创建模型库工作目录的第一级子目录。
* version: 是网元真实的版本号，**切记：要求与同目录下文件夹同名**。
* build：补丁号，或者叫做构建号，仅在更新现有网元版本，填写此字段。格式直接就是一个约定好的数字，例如6000，数字用于比较补丁号的大小。补丁号对于网元版本是唯一的，跟着version走，打补丁规则理论上是从小到大，不允许打完大补丁号的补丁后再打小补丁号的补丁，每一次补丁都会和基本网元版本进行合并，并在界面上显示出来，并可在**去激活**的状态进行删除。
* displayName: 仅在新增网元版本时，填写网元显示版本号，用于客户端界面呈现。
* remark：备注字段，对模型包做简明扼要的描述，用于对应下面的国际化信息字段。
* visible：此字段主要是对是否在界面显示此网元版本而设定的。如果是true，将会在界面上显示出来；如果是false，将不会再界面上显示出来。
* plugin：此字段标识是不是网元的工具（这个可以补充说明说明）

**框架支持一个zip包中包含多个网元版本模型。**

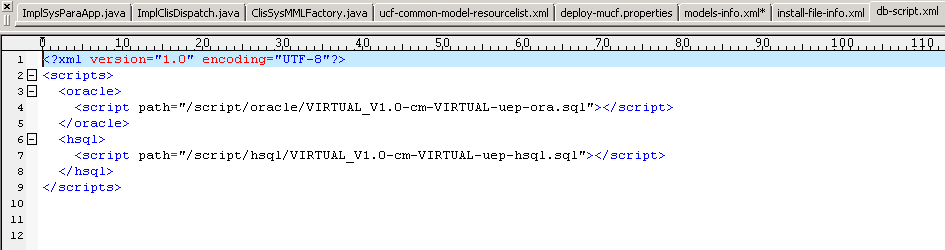
每个网元版本文件下面的结构如下图所示：



这里面每个文件夹对应一个模块版本号（这个模块版本和yang文件里的module字段相关），还有一个version文件，version文件内部描述拥有的模块版本信息，如下截图：



通常，MO模型包是不需要包含数据库脚本的（系统会自动根据mo模型文件维护数据库表结构及字段），除非产品有特殊需求，可指定执行产品定制的数据库脚本，脚本文件放置在script目录下，并编写脚本执行描述文件db-script.xml：



### 模型库的工作目录

### 虚拟网元版本技术

模型库管理的均是一个个网元版本（或制式版本）。比如对配置模型包而言，大部分网元逻辑都放在制式版本目录中，少量网元逻辑放在网元版本目录中。但是配置中还有一类变化比较频繁的业务逻辑，比如模版算法包，既不属于制式版本，也不属于网元版本。此时我们可以建立一个虚拟网元版本，将这些算法包放在这个虚拟网元版本目录下。**这个就是上述models-info.xml文件中，设置plugin字段的用处所在**。这样带来很多好处：

1. 可以借助模型库的导入，分发，加载，回退功能以及客户端分发功能，实现算法包的更新需求。
2. 借助模型网元版本号，补丁号技术，实现新旧算法包的替换覆盖管理。

这实际上借助模型包管理功能实现了网管功能的更新。

还有一类配置文件也可以使用虚拟网元版本技术进行网管分发，比如扁平化功能的开关项。在只读模式时，某些菜单项和服务需要关闭，而在全模式下，这些菜单项和服务需要打开。我们可以把这些开关项做成配置文件，放在一个虚拟网元版本下。这样通过导入模型包就可以实现开关配置文件的添加和更新。特别是对客户端也可以实现内容的分发和推送，从而避免为每个客户端打补丁。

对于告警领域，它的模型包只有一套最新的版本，向下兼容所有网元版本，因此告警的模型包版本也可以是一个虚拟网元版本号，只有虚拟网元版本号比较新，或者补丁号比较大，就可以实现新包覆盖旧模型的文件。

## 模型包的特征和分类

### 模型包版本标识

模型库按网元版本号来区分，不同网元版本号代表不同的网元版本。同一个网元版本号，可以打多次补丁，补丁用网元版本号后面的build来标识。补丁号不作为关键字，只用于描述当前的补丁信息。因此给网元模型版本打补丁并不涉及升级网元版本。

网元版本号和补丁号均由版本经理统一管理，对于补丁号，各个领域模块制作补丁前，均需要向版本经理申请补丁编号，然后各自制作相关补丁。**最终生成的补丁包中，各领域模块中可能会包含多个补丁号，系统会将每一次的打入系统的补丁都显示在其对应的网元版本的下方，系统会自动将刚刚导入的补丁和系统中当前存在的最新对应的网元版本进行合并，此时合并之后的网元版本应该包含基本版本+历史补丁+最新补丁**。

模型库打补丁不支持像网管补丁那样的紧急补丁（即通知单）。因此模型库补丁最好是提前规划好，按计划提供。

### 模型包补丁号规则

同版本的模型包，如果版本号补丁号都相同，则不允许在网管中导入生效，为了解决这个问题，可以利用更新模型包的补丁号来解决，当导入的模型包，补丁号更大时，系统允许导入。

整个网管开发是根据统一的迭代号来规划的，每个迭代都可以发布一个网元模型包，为了便于测试和使用，可以在补丁号中增加迭代号信息，这样每次发布的模型包都是更大的补丁号，因此可以直接导入生效。比如S59迭代，则模型包的补丁号为build=5900。后面2个0是为了方便出真正的补丁的编号。

### 模型包和网管补丁的关系

模型包和网管补丁分离，也就是说模型包的内容不在网管补丁中体现，它和网管补丁的内容是正交的。这样后续模型包更新和网管补丁更新就不会冲突。

### 模型包详细分类（暂时不考虑）

## 模型库的使用

## 各领域模块的模型定义

### 配置模型信息分类

### 配置模型包目录设定建议

## 模型初始化默认激活（**在扁平化时代进行的策略**）

### 概述

要实现模型包在ems安装时自动激活对应的制式模型，需要满足2个条件：

1、各个制式的产品提供制式标识文件。用于表示当前网管支持哪些制式。

2、对每个网元模型，包括虚拟网元模型，标识在什么制式下需要被激活。

模型包在ems安装时自动激活原理：

每次安装或增量安装ems时，uep都会把所有模型包拷贝到特定目录。Ems第1次启动后，模型管理进程首先检测特定目录中是否有模型文件，如果有就按照安装的制式信息进行激活，（已激活的模型包会跳过）。最终删除特定目录中的模型文件。下次ems重启时就不会再次激活了。

这样网管在大版本发布时，内部就已经提供了大版本对应的网元模型文件。升级网管时自动增加新的网元模型。而网管中网元模型管理的功能，则主要应对大版本之间的一些模型补丁，比如新增一款rru的热补丁。

### 标识文件

这个文件是由产品提供，放在各个制式的安装包中，在安装ems的时候放置到指定的目录文件夹下面。这个文件可以没有内容，仅仅用文件名进行标识。文件名规范：

UMTS.tag

GSM.tag

TDS.tag

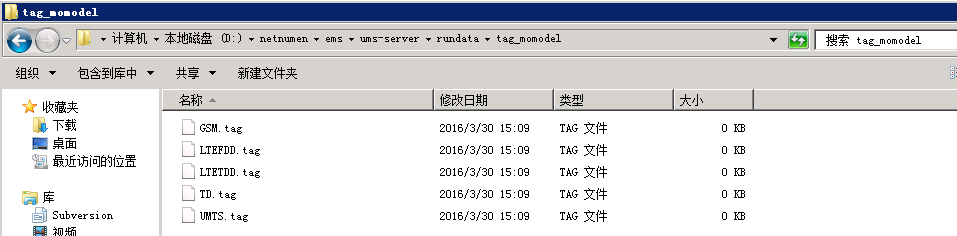
CDMA.tag

LTEFDD.tag

LTETDD.tag

文件放置路径：ums-server\rundata\tag\_momodel\

样式截图：



tag文件的目的是标记这个ems安装并支持了哪些制式。tag文件应该直接放在制式主安装包的zip文件中。

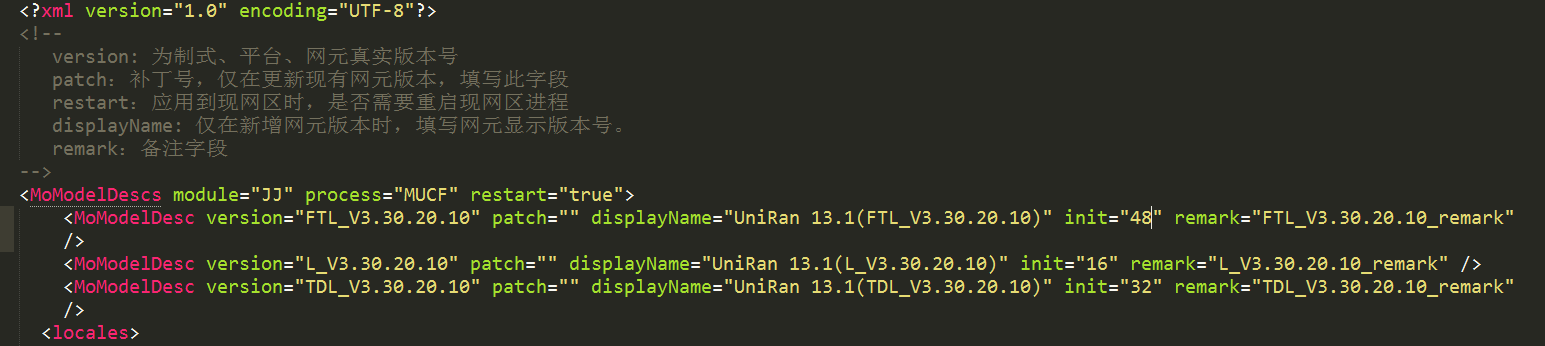
举例来说：LTEFDD.tag这个文件应该放在fddV3-V12.15.10B56.zip文件中，目录就是ums-server\rundata\tag\_momodel。 ems在安装时就会自动将这个tag文件拷贝到对应的目录位置。

如果针对ltefdd制式，产品提供了多个zip安装包（比如还有一些ltefdd工具安装包，北向安装包），那么tag文件只需要放在最主要的安装包中就可以了。因为要支持某个制式，主安装包是必选的，主安装包选中安装了，就认为ems支持了这个制式。

### 模型包内部修改

对模型包内部的修改，主要作用是将模型包内部的具体产品模型文件与安装了的产品相关联起来，以此来进行默认激活模型。需要产品在\*-models-info.xml文件中添加init属性。

建议配置文件修改截图：



Init中填写制式对应的数字值（1->UMTS，2->GSM，4->TDS，8->CDMA，16->LTEFDD，32->LTETDD），累加表示多模，逗号表示或的意思。

Init=”1” 表示当安装了umts制式时激活此模型包

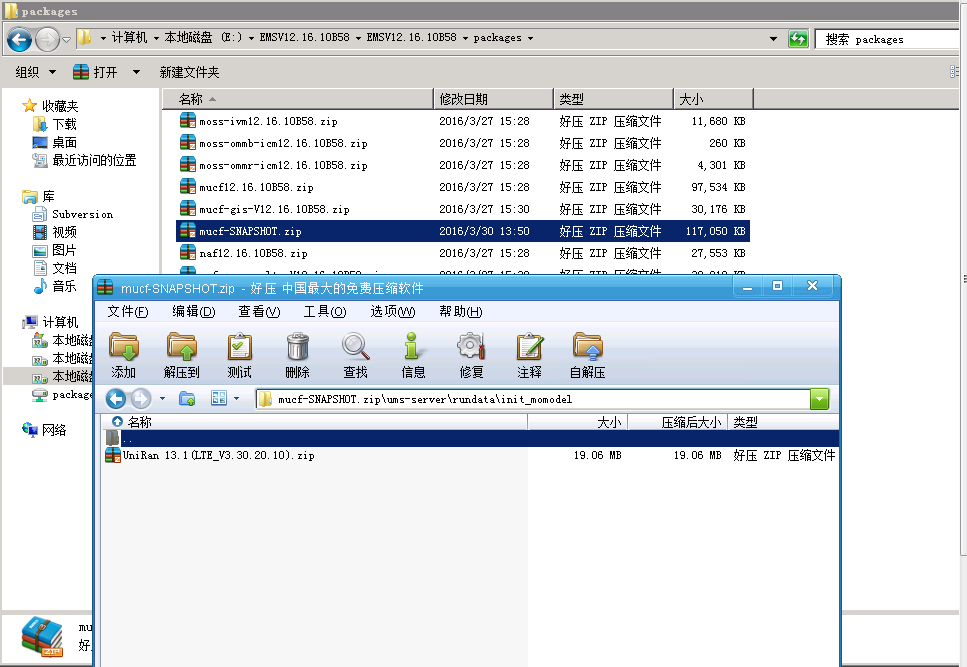
Init=”3” 表示当安装了umts**和**gsm制式时激活此模型包(必须同时安装gsm和umts，一般是多模网元模型包)

Init=”1,2” 表示当安装了umts**或**gsm制式时激活此模型包

如果没有init字段，则表示无论安装什么制式都会激活该模型包

### 初始化模型包的放置路径

初始化模型包可以作为一些安装包放在ems的package中。即xxx网元模型安装包.zip，在zip文件中放置目前的网元模型包，目录为：\ums-server\rundata\init\_momodel\xxx网元模型包.zip。可以参照如下截图：



网元模型包继续使用原来的模型包zip文件，其中LTE模型包提供plat+lte的网元模型，TL模型包提供TDS+TL双模的网元模型。

安装包的配置信息文件zipPackageName.xml，需要注意如下事项：

1、必须设定不可见，但是默认安装。id必须固定，后续不能修改。

**<zip id="ums.install.production.ne\_model.lte" visiable="false" defaultInstall="true">** 即不可见，但是默认安装。Id必须固定，后续不能再修改。

**对lte的模型安装包，id为"ums.install.production.ne\_model.lte"**

**对TL的模型安装包，id为"ums.install.production.ne\_model.tds"**

**对GU的模型安装包，id为"ums.install.production.ne\_model.gsm\_umts"**

**对cdma的模型安装包，id为"ums.install.production.ne\_model.cdma"**

2、在扁平化切换开关没有做好之前，先把zipPackageName.xml改成zipPackageName.xml.bak

以lte的模型包为例：

**<?xml version="1.0" encoding="GBK"?>**

**<zip id="ums.install.production.ne\_model.lte" visiable="false" defaultInstall="true">**

**<!-- 该zip文件的描述 -->**

**<description>**

**<locale en="lte Model Package" />**

**<locale zh="LTE模型包" />**

**</description>**

**<!-- 该zip文件所属的ppu，若一个ppu对应多个zip文件，那么这儿定义的描述最好为一致的，如果不一致则采取最先一个zip包的描述 -->**

**<owner-ppu id="MCM">**

**<description>**

**<locale en="SDR Base Station Configuration Management" />**

**<locale zh="SDR基站配置管理" />**

**</description>**

**</owner-ppu>**

**<!-- zip文件依赖的其他的zip文件，可依赖多个zip文件，该依赖决定了后面的脚本执行顺序。 -->**

**<depend>**

**<zip id="ums.install.production.uca" />**

**</depend>**

**<internaldbs>**

**</internaldbs>**

**</zip>**

## 模型库激活对各个模块消息通知接口，与消息体结构