目录

[引言 2](#_Toc507663009)

[一、上市保障日常工作 2](#_Toc507663010)

[二、上市保障痛点 2](#_Toc507663011)

[三、解决方案 2](#_Toc507663012)

[四、工具使用示例 3](#_Toc507663013)

[1.流程梳理 3](#_Toc507663014)

[2.事务跟踪在TaskSteper系统中的实现(管理员部分) 3](#_Toc507663015)

[2.1)定义字段Model 3](#_Toc507663016)

[2.2)定义字段Form 4](#_Toc507663017)

[2.3)将上述两个步骤定义的Model和Form添加到FormAndModelDict中。 4](#_Toc507663018)

[2.4）将定义好的model迁移到数据库中 4](#_Toc507663019)

[2.5）定义状态转换表FSM\_TRANS\_ISSUE\_TRACK 6](#_Toc507663020)

[3.用户使用TaskSteper跟踪问题处理（用户部分） 6](#_Toc507663021)

[五、其他案例 9](#_Toc507663022)

[1.搭建多个项目统一的版本记录平台 9](#_Toc507663023)

[六、进一步优化 11](#_Toc507663024)

# 引言

日常工作包含众多事项，每件事项具有多个环节，每个环节需要不同的角色来处理，怎样将**事项、环节、员工角色**三者系统地组织起来；既能有效的跟踪每项事务落地，又能最大程度降低沟通交流成本？

本文档以上市保障为例，介绍一种利用工作流软件管理日常工作的方法；除上市保障外，利用工作流软件管理其他日常工作的方法与之类似。

# 一、上市保障日常工作

上市保障工作开始意味着本轮开发工作基本结束，产品上市后的运行状况是对产品开发质量的一次检验；上市保障工作除及时解决客户现场暴出的问题外，一个重要的职责是从在网设备问题中提取避免问题重犯的准则和改进点，并促使这些准则和改进在下一轮的开发中落地以提升新一代产品质量。

上市保障工作可分为主动维护和被动维护两大方面，每个方面又可以细分为多项具体工作，如下所示：

* 被动维护：主要指问题发生后的响应和处理；如上市保障问题处理，事故的快速恢复，ITR、投诉的快速关闭，问题发生后的客户情绪安抚等。
* 主动维护：主要指避免问题发生，提升产品质量、DFX特性，提高客户满意度等；如针对金融、国家电网等客户设立重点局设备档案；阿里等重点客户在双十一等关键时间点的保障；避免处理问题过程中发现的低级错误重犯；高效地跟踪改进点落地；快速将典型问题的处理技能传递给轮转到上市保障组的同事等。



# 二、上市保障痛点

* 工作内容复杂，耗费大量精力依然无序。
* 被问题攻关搞得焦头烂额，彻夜不眠，最后发现是一个低级问题
* 屡次被客户质疑，因为产品质量羞愧难当。
* 解决这些问题的过程中有很多闪亮的点子，一些可供后续设计、开发时参考进而避免重蹈覆辙的规范，可惜没有积累下来，新一代产品开发时又在问题的泥潭中苦苦挣扎。

怎样系统有序的管理各项工作？怎样避免初级问题重犯？上市保障的典型问题如何积累下来，怎样对上市保障组的新组员进行快速赋能？

# 三、解决方案

* 1.梳理日常工作，将例行工作划分为多个业务流程，如问题处理流程，改进流程等；
* 2.每个业务划分多个环节，并为每个环节指定处理人；
* 3.规划好以后由专人利用业务流程管理(BPM)工具进行实施；
* 4.工具实施后，各角色作为用户登录系统，处理分配给自己的问题单，处理完成后将问题单走给下个环节处理人即可；如问题处理流程中问题处理人完成问题分析，维护代表完成分析审核，SE完成改进分析并创建改进问题单等。
* 在处理问题的过程中，完成典型问题的提取、优化改进的提取等工作，后续可以登录系统查看跟踪优化改进的落地，也可以查看归档的典型问题。

TaskSteper是基于django框架实现的建议的BPM工具，可以协助各任务角色高效地分工协作。

# 四、工具使用示例

## 1.流程梳理

将上市保障组的日常工作分为网上问题处理、重点局档案管理、负向改进、典型Buglist管理等几个流程；并为每个流程划分为多个环节，以流程确保被跟踪的事务闭环，如下所示：



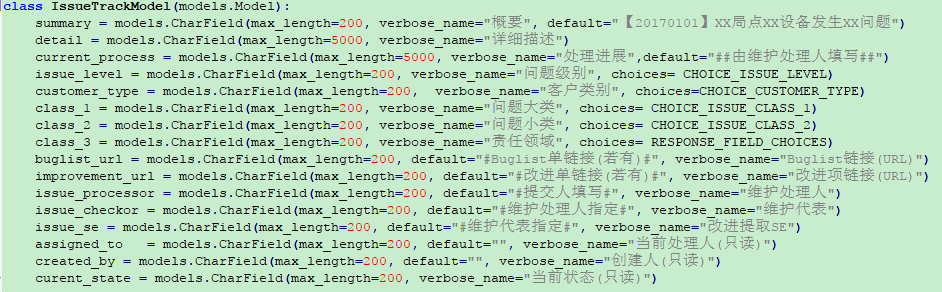
## 2.事务跟踪在TaskSteper系统中的实现(管理员部分)

该部分属于简单的二次开发，部门中有一个人熟悉工具即可，跳过该部分不影响对全篇文档的理解，可直接阅读“**用户使用TaskSteper跟踪问题处理（用户部分）**”；

以网上处理流程为例，步骤如下：

### 2.1)定义字段Model

此处定义为IssueTraceModel，一个Model对应数据库的一张表，Model的一个field，如summary、detail等对应表中的一列。

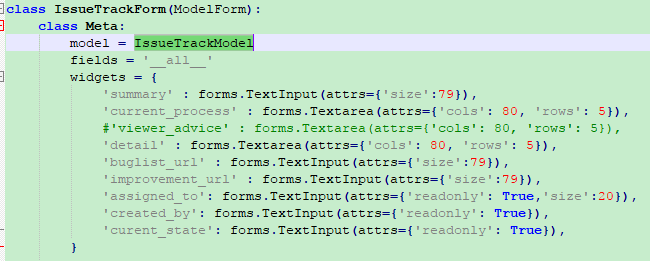


Field的属性说明如下：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | max\_length | 字段的最大长度 | | verbose\_name | 字段在界面的显示名称，见下图 | | default | 字段默认值 | | choices | 下拉型字段的选择列表，class\_1字段的choices如右图所示，需在IssueTraceModel之前定义。 | |  |

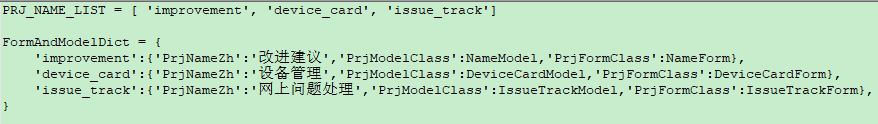
### 2.2)定义字段Form

Form对应于界面显示一个表单，fields属性决定Model中的哪些字段会显示在表单中（‘\_\_all\_\_’表示全部显示），widgets主要定义了字段在显示时的一些属性，如显示的大小，是否只读等。此外需要将Form的model属性设置成步骤2.1中的model。



### 2.3)将上述两个步骤定义的Model和Form添加到FormAndModelDict中。

TaskSteper支持创建多个项目，每个项目有独立的流程和字段，FormAndModelDict中’ issue\_track’为项目名,’网上问题处理’为网页上显示的项目中文名。PrjModelClass和PrjFormClass分别指定每个项目使用的Model和Form类。



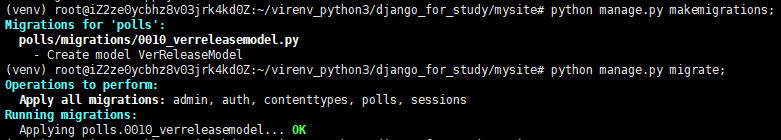
### 2.4）将定义好的model迁移到数据库中

在django工程目录下执行以下两条命令即可：

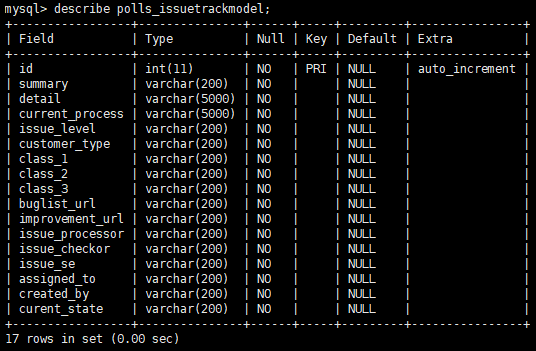
python manage.py makemigrations;

python manage.py migrate;

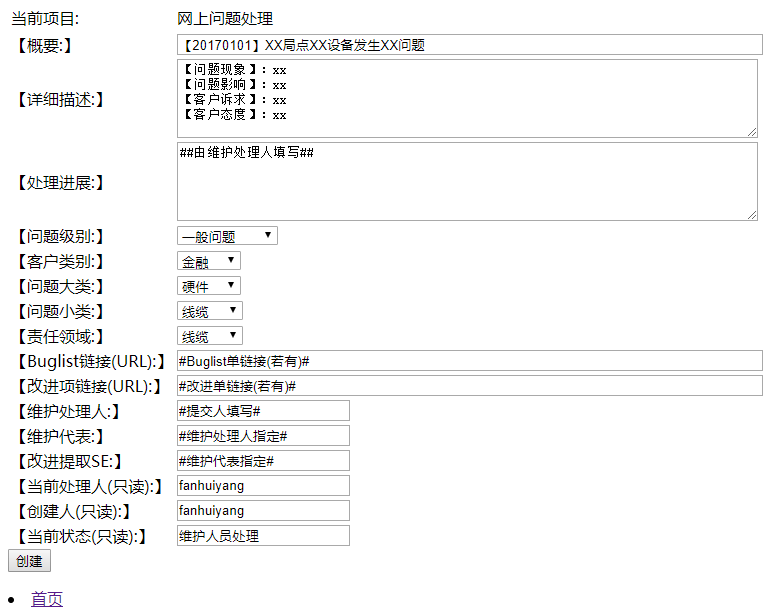
迁移成功后会看到类似如下提示（后补示意截图，其中的model名字与上文要增加的model不一致，可不关注）：



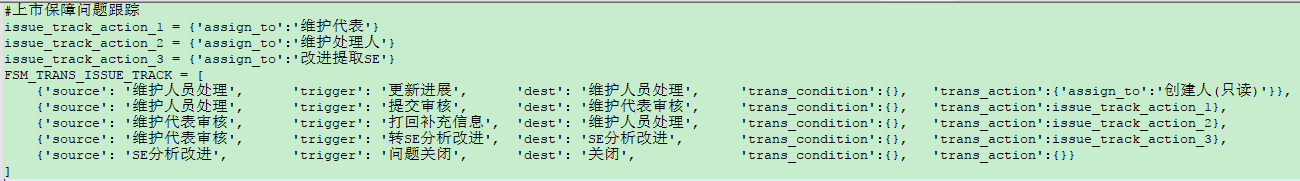
迁移完成后可以看到数据库中已经创建了与Model对应的数据库表：



通过uwsgi启动django服务，可以看到定义完成后界面显示效果如下所示：



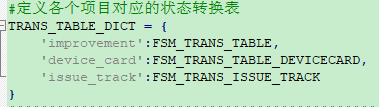
### 2.5）定义状态转换表FSM\_TRANS\_ISSUE\_TRACK



此处一个状态对应日常工作的一个环节，表中每一行表示一个状态转换，包括以下元素：

|  |  |
| --- | --- |
| source | 源状态 |
| trigger | 对应界面上的触发按钮，点击按钮时会将问题单转到目标状态 |
| dest | 目标状态 |
| trans\_condition | 转换条件，只有满足转换条件才会生成相应的按钮 |
| trans\_action | 转换时的操作，目前支持assign\_to（将问题分配给某个角色）等 |

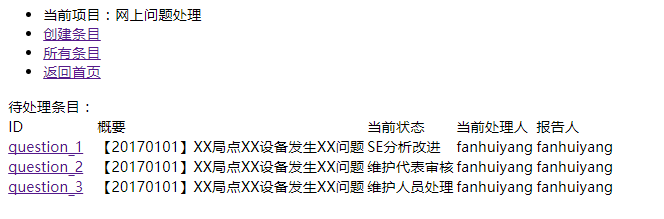
定义完成后将项目名称和对应的状态转换表添加到TRANS\_TABLE\_DICT中：

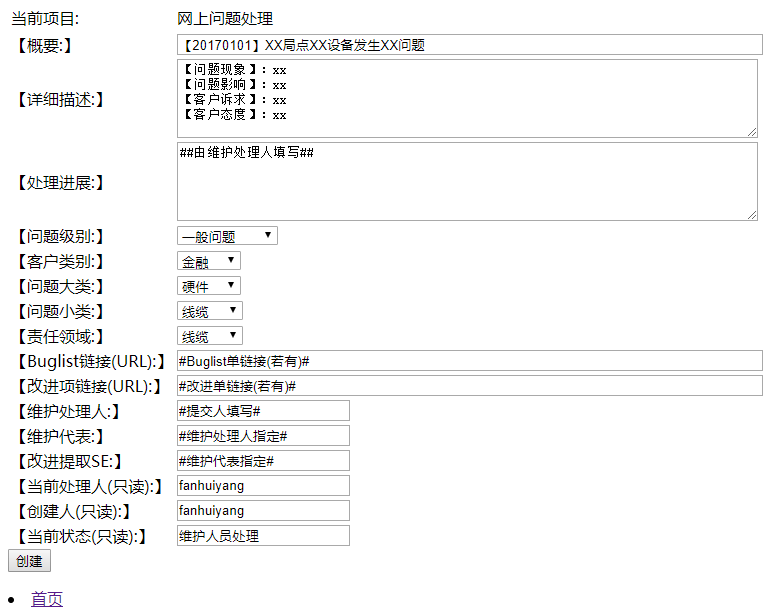


## 3.用户使用TaskSteper跟踪问题处理（用户部分）

**具体到一个问题的处理，流程如下所示**

* 维护人员接收到问题后创建条目，录入问题单库，记录时间、局点、设备型号、问题描述等信息；

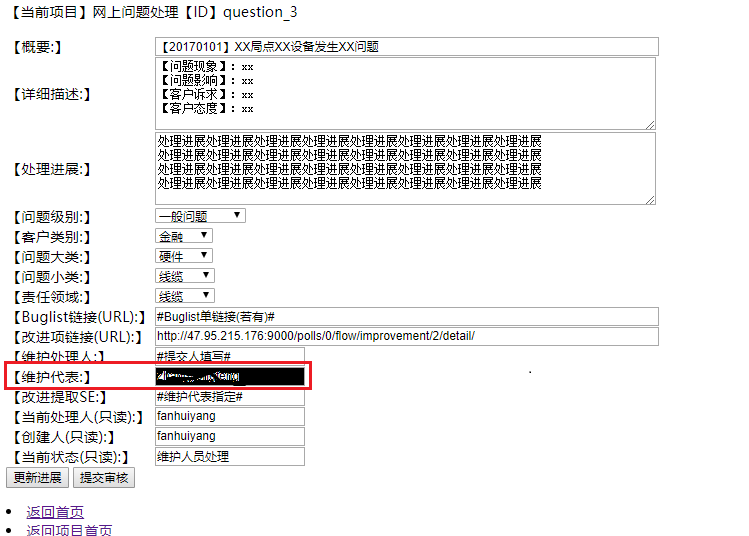




* 在“维护人员处理”阶段，问题处理人补充问题信息，更新问题进展；

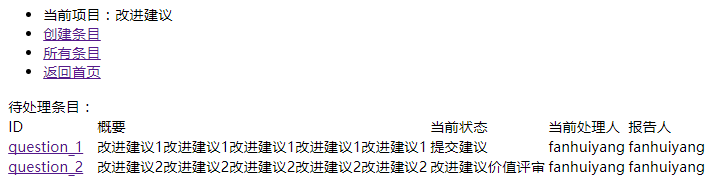
如果该问题是多次出现的典型问题，需要在“buglist”管理项目中增加一个问题，并将问题地址填入此处的“Buglist连接(URL)”字段中。

处理完成后填入维护代表，点击“提交审核”将问题单提交审核，问题转给下个角色（维护代表）处理。

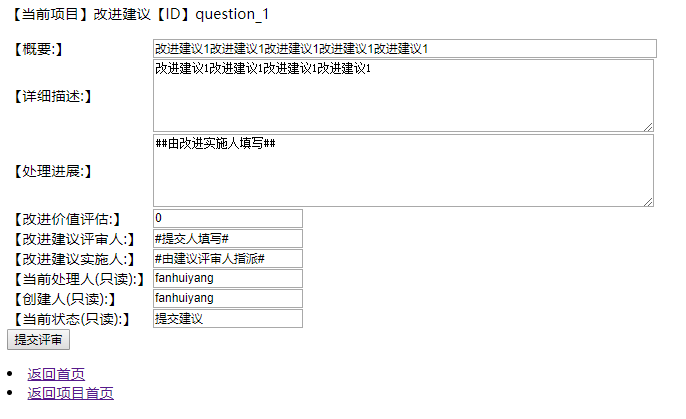


* SE分析改进阶段，若从该问题中发现需改进点，则创建一个改进单用于跟踪改进落地，并将改进单的地址填入“改进项链接(URL)”字段。网上问题处理完成后即可关闭，发现的改进点通过改进单跟踪落地。后续再次查看该网上问题时，可以根据“改进项链接(URL)”字段跳转到改进单地址，了解该问题提出过哪些改进点。

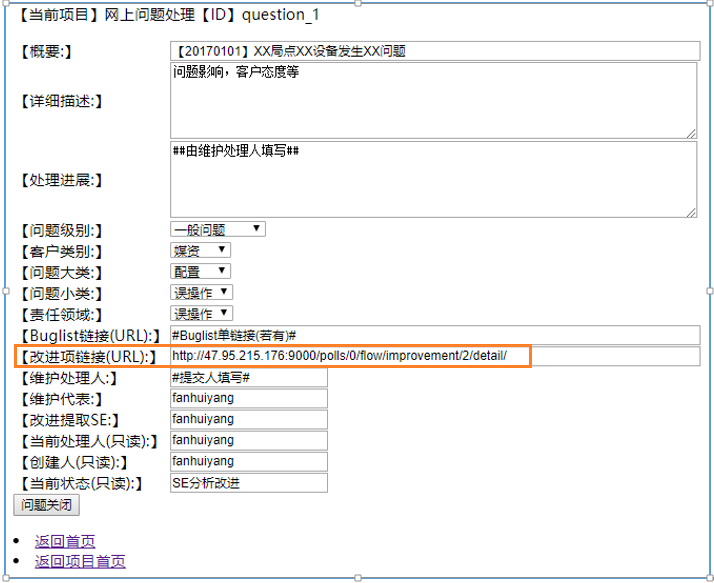
创建的改进建议列表示例：



创建的改进项详细信息示例：

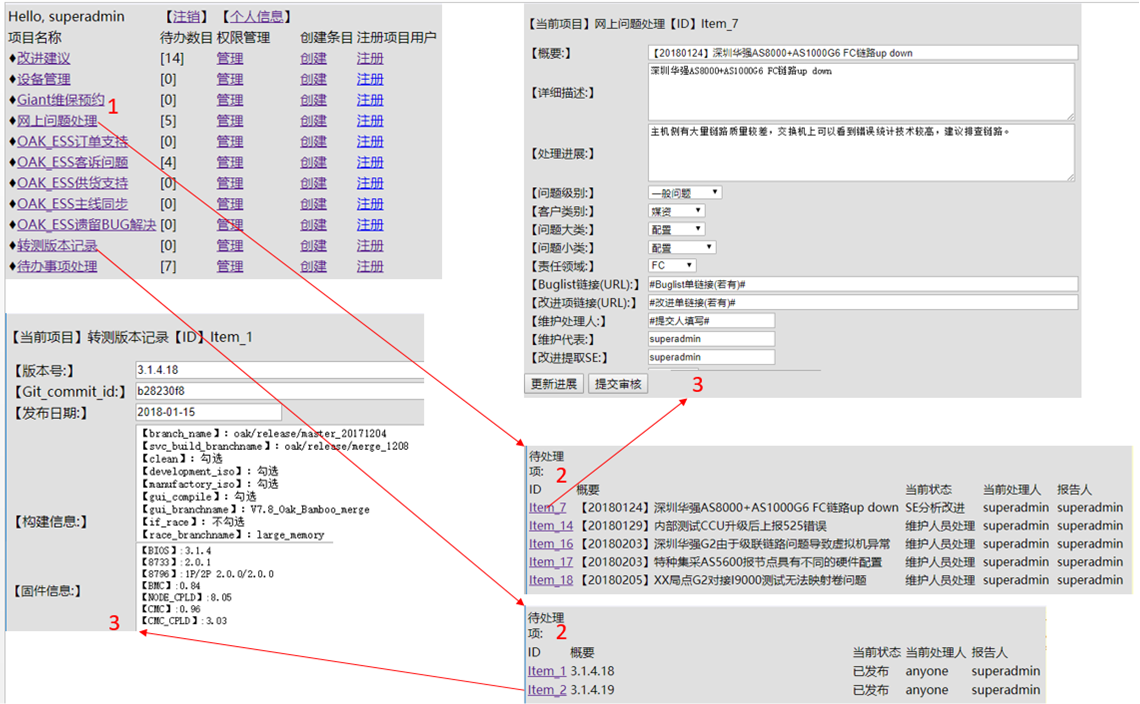


跟踪网上问题处理的问题单中，利用“改进项链接(URL)”字段记录改进单的地址，在网上问题处理和改进建议跟踪落地两个项目间建立联系，如下所示：



利用该工具可以搭建集成的工作平台，类似现在使用的wiki记录了各种各样的静态信息，该工具可以用于跟踪可流程化的各种事务，如网上问题从接收到闭环，改进建议从收集到跟踪落地，各个版本的发布等。

下图用于跟踪网上问题处理和记录发布版本信息的一个例子，供参考。页1是创建的各个项目列表；页2是每个项目下的问题列表；页3是各项目下问题详细信息；工具可以根据需求定制流程、添加字段，比如用于问题分类的字段、问题级别、客户类型等。



使用该工具的优势：

1. 极大降低沟通成本

要了解一项工作的进展，只需要登录系统找到表示该工作的问题单就能一目了然，避免了表格跟踪事务时所需的专人拉会收集信息、汇总、分发、各人更新等繁琐的步骤。

1. 有效跟踪事务闭环

各环节责任人明确，当前状态明确，可以有效地跟踪事务从开始到闭环的全过程。

1. 团队知识有序的积累和归档

日常工作按照上述方法开展，在问题处理过程中就可以完成典型问题的积累，改进点的提取、跟踪、落地等，并且在问题处理项目和Buglist、改进流程之间通过URL链接建立联系，可以通过问题模块、发生局点等信息等快速找到相应问题单，进而找到积累的典型问题信息或者改进信息；也可以通过典型问题信息或者改进信息找到对应的网上问题单，了解问题发生背景；是一种较好的团队知识归档方法。

# 五、其他案例

## 1.搭建多个项目统一的版本记录平台

**背景和痛点**：

当前X行业集采验证过程中，遇到一个问题需要搭建LDBE环境解析抓取的内存dump二进制文件进行分析，搭建环境需要获取对应版本的/home/debug目录（其中包含解析二进制所需的符号表等多个文件）。

目前缺乏统一的版本记录平台（CQ平台太封闭，并非人人有权限），要获取历史版本的/home/debug需要找到负责构建这个分支的同事，回退代码版本，填写若干jenkins参数重新编译；

找人、人员沟通、参数确认、构建等工作耗费时间过多，很容易大半天就过去了，而且可能构建分支已经更新，导致编译出的/home/debug无法解析dump文件；

更严重的是在网设备运行一两年后，出现网上问题需要分析dump文件，因时间过长版本修改过大、人员变动等回退到发货时的版本重新构建/home/debug将更为困难，很有可能在客户要求的故障恢复时间窗内连分析dump文件所需的环境都搭建不好。

**期望达到的效果:**

在一个统一平台记录好发布版本相关的若干信息，并保存安装iso、gpg包、/home/debug文件包等，需要时直接取用，人均耗时可控制在5分钟内。

**解决措施：**

下图是自己记录的OAK项目版本信息，供参考：

每发布一个版本时填写【版本号】、【git\_commit\_id】、【发布日期】等信息，【构建信息】、【固件信息】等可以从上一个版本记录直接复制粘贴再进行相应改动，记录这些信息不需要多少时间；但是可以节约大量后期用于追溯版本信息的时间。

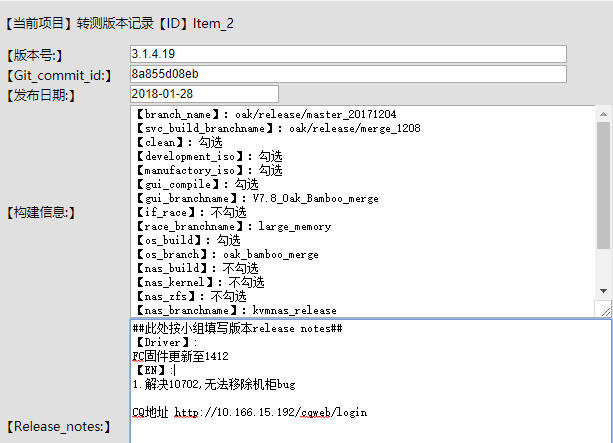
可参照上述OAK版本信息记录工具，设立一个统一的版本记录平台用于记录所有项目的发布版本。可以添加专门的字段用于区分项目，也可通过在概要中添加后缀进行区分，如“3.1.4.19-OAK”，“2.1.0.1-TZ”等，；

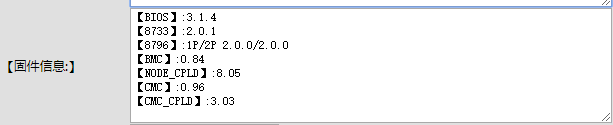
对于系统iso，gpg包，/home/debug文件包等较大的文件可以放在专门的ftp上，在版本记录平台中增加【系统安装包URL】等字段记录ftp地址，需要时可以直接跳转过去取用，提高效率。

**<a>记录的版本列表：**

cid:image001.jpg@01D3B07F.AB0BF8A0

**<b>某个版本的详细信息：**





# 六、进一步优化

增加按字段搜索、统计、导出等功能。

例如现在每次发版本需要手动写release notes，实际上可以增加“release notes”字段，每次发版本时，根据问题单走到解决状态的时间、技术小组等过滤出这一版本解决问题，然后将这些问题的“release notes”字段导出即可。