|  |
| --- |
| **FCM- Parameter** |

|  |  |  |
| --- | --- | --- |
| Date  日期 | Version  版本 | Comments  备注 |
|  | .0.1 | Creation of Document |
|  | .0.2 | Add devices.xml and partition part |
|  | .0.3 | Update for ro.product.\* and fingerprint |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Function  职位 | Name  姓名 | Date  日期 | Signature  签名 |
| Written by  拟定 | **System engineer** | **Liu Yongfeng** | 2018-12-20 | **Liu Yongfeng** |
| Written by  拟定 | **Integration engineer** | **Fan Yi** | 2019-12-30 | **Fan Yi** |
| Written by  拟定 | **APP engineer** | **Wang Xiongke** | 2019-1-10 | **Wang Xiongke** |
| Verified by  审核 |  |  |  |  |
| Verified by  审核 |  |  |  |  |
| Approved by  批准 |  |  |  |  |

Contents

[1.工厂参数介绍 2](#_Toc24241)

[1.1. FCM额外参数在traceability分区定义 2](#_Toc8705)

[1.1.2. MTK平台 2](#_Toc13399)

[1.2. FCM参数关系和说明 2](#_Toc15480)

[1.2.1. ECID 2](#_Toc10970)

[1.2.2. Variant 3](#_Toc16082)

[1.2.3. Subvariant 3](#_Toc19265)

[1.2.4. Model name 3](#_Toc23518)

[1.2.5. Product name 3](#_Toc7031)

[1.2.6. Product 3](#_Toc3790)

[1.2.7. Brand和market name 3](#_Toc19215)

[1.3. 举例说明（以A5x项目为例） 4](#_Toc3436)

[2. Devices.xml介绍 5](#_Toc29482)

[3. Partition Table介绍 5](#_Toc17637)

[3.1.scatter文件生成 5](#_Toc27268)

[3.2. partition table分区参数更新 7](#_Toc32293)

[3.3. 修改partition table 7](#_Toc30454)

[3.3.1. 添加devices.xml 7](#_Toc28119)

[3.3.2. 修改分区csv文件 7](#_Toc12266)

[3.3.3. 修改ptgen脚本 7](#_Toc21883)

[4. build property生成 9](#_Toc16792)

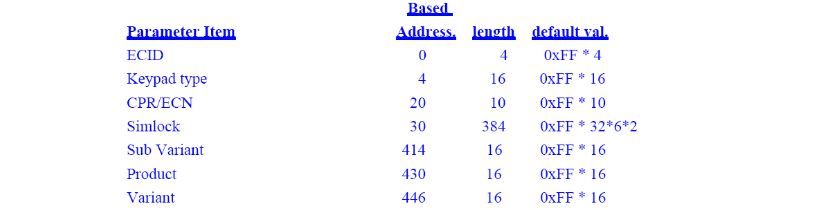
[5. fingerprint生成 9](#_Toc17982)

[6. 根据ecid配置system property 10](#_Toc32518)

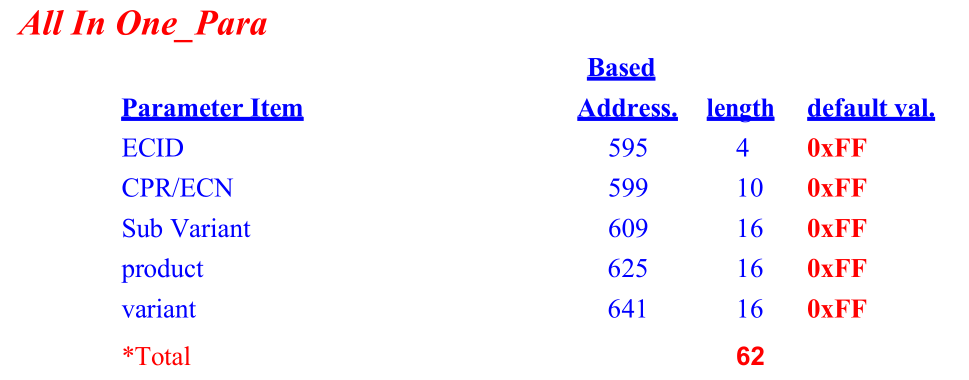
此文档主要介绍FCM(Flexible Customization Mechanism)方案相关参数。

# 1.工厂参数介绍

1.1. FCM额外参数在traceability分区定义

1.1.1. 高通平台

1.1.2. MTK平台



1.2. FCM参数关系和说明

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Boot Properity** | **App properity** | **Related Parameter** |
| **ECID** | **ro.boot.ecid** | **ro.ecid** |  |
| **subvariant** | **ro.boot.subvariant** |  |  |
| **Product** | **ro.boot.product** |  |  |
| **variant** | **ro.boot.variant** |  |  |
| **Model name** | **ro.boot.model** | **ro.product.model** | **CU** |
| **Product name** | **ro.boot.name** | **ro.prodcut.name** | **model+variant+subvariant** |
| **brand** | **ro.boot.brand** | **ro.product.brand** | **model** |
| **market** | **ro.boot.publicname** | **ro.product.publicname** | **model** |

1.2.1. ECID

* **ECID相关介绍和定义见《FCM ECID naming rule.ppt》文档**
* **ECID由工厂生产时注入到traceability分区，具体位置参照上面分区定义**
* **LK通过cmdline ro.androidboot.ecid转换成ro.boot.ecid property**

1.2.2. Variant

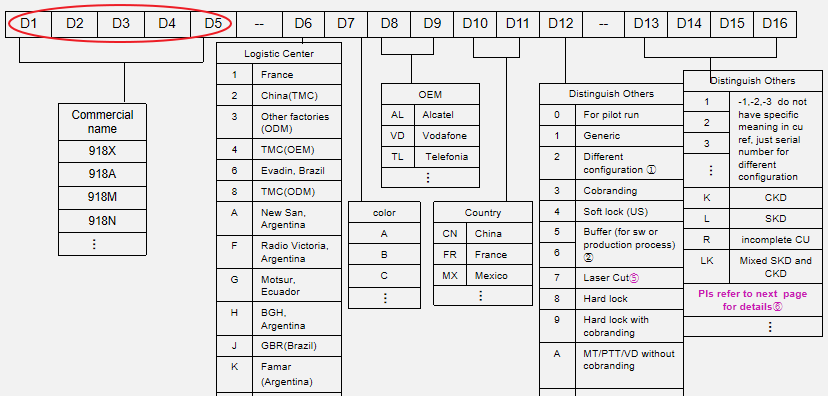
* **项目组通过codename定义，由工厂生产注入traceability分区，具体位置参照上面分区定义**
* **LK通过cmdline ro.androidboot.variant转换成ro.boot.variant property**

1.2.3. Subvariant

* **项目组通过codename定义，由工厂生产注入traceability分区，具体位置参照上面分区定义**
* **LK通过cmdline ro.androidboot.subvariant转换成ro.boot.subvariant property**

1.2.4. Model name

* **CU定义如下，在工厂生产时注入，LK读取前几位即为Model name**
* **LK通过cmdline ro.androidboot.model转换成ro.boot.model property**



1.2.5. Product name

**由model，variant和subvariant拼接而成，即**ro .androidboot.name=**model+variant+subvariant，转换成ro.boot.name property**

1.2.6. Product

* **工厂生产时注入traceability分区，具体位置参照上面分区定义**
* **LK通过cmdline ro.androidboot.product转换成ro.boot.product property**

1.2.7. Brand和market name

**如下定义struct（由devices.xml转换生成），通过model/code name mapping brand和marketing name，ro.boot.brand通过LK cmdline转换而来，****ro.boot.publicname由于有空格，在property\_service.cpp文件中处理**





1.3. 举例说明（以A5x项目为例）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Brand** | **Marketing Name** | **Product** | **Variant** | **SubVariant** | **Code Name** |
| **Alcatel** | **Alcatel 5V** | **5060** | **emea** |  | **5060D** |
| **emea** | **ru** | **5060D** |
| **latam** |  | **5060A** |
| **India** |  | **5060J** |
| **IRE** | **IRE 8** | **brazil** |  | **5060K** |

黄色行对应的property如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ro.boot.variant** | **ro.boot.subvariant** | **ro.boot.name** | **ro.boot.model** | **ro.boot.brand** | **ro.boot.pulicname** |
| **emea** | **ru** | **5060Demearu** | **5060D** | **Alcatel** | **Alcatel 5V** |

LK部分开发代码请参考[vendor](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/)/[mediatek](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/)/[proprietary](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/)/[bootable](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/bootable/)/[bootloader](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/bootable/bootloader/)/[lk](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/bootable/bootloader/lk/)/[app](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/bootable/bootloader/lk/app/)/[mt\_boot](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/bootable/bootloader/lk/app/mt_boot/)/[mt\_boot.c](http://opengrok.tcl-mobile.com/A5X_P_All-in-One/xref/vendor/mediatek/proprietary/bootable/bootloader/lk/app/mt_boot/mt_boot.c)文件

# Devices.xml介绍

代码路径：device/mediateksample/a5x/devices.xml

**<?xml version="1.0" encoding="utf-8"?>**

**<devices>**

**<device name="5060Demea" product="5060" model="5060D" modem\_tag="emea" oem\_tag="common" svnumber="1" oem\_code="EU" brand="Alcatel" marketing\_name="Alcatel 5V"/>**

**</devices>**

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **name** | **codename + variant + subvariant** |
| **product** | **product code** |
| **model** | **code name** |
| **modem\_tag** | **variant** |
| **oem\_tag** | **subvariant** |
| **svnnumber** | **product SVN number** |
| **oem\_code** | **download code for multi oem, modem and logo images** |
| **brand** | product branch |
| **marketing\_name** | product marketing name |

# Partition Table介绍

3.1.scatter文件生成

**通过devices.xml转换如下信息写入partition table: MTxxxx\_Android\_scatter.txt**

|  |
| --- |
| **###################################################################################**  **# Variant Infos**  **###################################################################################**  **- device\_index: 01**  **name: 5060Demea**  **product: 5060**  **model: 5060D**  **modem\_tag: emea**  **oem\_tag: common**  **svnumber: 01**  **oem\_code: ZZ** |

**oem\_code : 对应 modem，logo以及 oem 镜像文件的 oem\_code , 和原 PERSO 定制框架下重命名 Perso 文件时的 命名方式相同, oem\_code 对应的两位也出现在 第 6,7 位. 所有这些文件和其他文件一起存放, 不会有单独目录.**

**如：**

**variant为“emea”，subvariant为空的product(5060)对应的oem\_code为“ZZ”，需要下载重命名后第6,7位为ZZ的对应的分区镜像文件。**

3.2. partition table分区参数更新

**分区“is\_oem”字段为“true”时，通过modem\_tag和oem\_tag匹配对应的“oem\_code”值。**

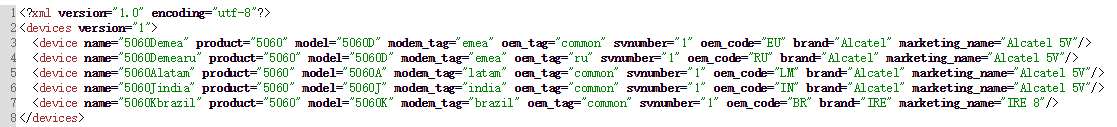
|  |
| --- |
| **-partition\_index: SYS32**  **rename\_prefix: C << 重命名后的分区镜像文件首字母**  **partition\_name: oem**  **file\_name: oem.img**  **is\_download: true**  **is\_oem: true << 是否为perso image。true: 有多个镜像文件，需根据variant/subvariant选择**  **type: EXT4\_IMG**  **linear\_start\_addr: 0x1de00000**  **physical\_start\_addr: 0x1de00000**  **partition\_size: 0x40200000**  **region: EMMC\_USER**  **storage: HW\_STORAGE\_EMMC**  **boundary\_check: true**  **is\_reserved: false**  **operation\_type: UPDATE**  **is\_upgradable: true**  **empty\_boot\_needed: false**  **reserve: 0x00** |

**oem分区(partition\_index: SYS32)“is\_oem”为“true”,即OEM为 perso image，有多个镜像文件选择下载；**

**rename\_prefix为“C”，OEM分区对应重命名后以“C”为首字母的镜像文件。**

3.3. 修改partition table

3.3.1. 添加devices.xml



3.3.2. 修改分区csv文件

* **partition\_table\_MTxxxx\_emmc.csv添加“Rename\_Prefix”列**
* **Rename\_Prefix的值对应各分区镜像文件rename后的首字母**

3.3.3. 修改ptgen脚本

|  |
| --- |
| **diff --git a/build/tools/ptgen/MT6765/ptgen.mk b/build/tools/ptgen/MT6765/ptgen.mk**  **index 6e4bb05..0c3e24a 100644**  **--- a/build/tools/ptgen/MT6765/ptgen.mk**  **+++ b/build/tools/ptgen/MT6765/ptgen.mk**  **@@ -58,6 +58,7 @@ MTK\_PTGEN\_COMMAND := \**  **SYSTEM\_AS\_ROOT=${SYSTEM\_AS\_ROOT} \**  **MTK\_BUILD\_ROOT=${MTK\_BUILD\_ROOT} \**  **MTK\_DTBO\_UPGRADE\_FROM\_ANDROID\_O=${MTK\_DTBO\_UPGRADE\_FROM\_ANDROID\_O} \**  **+ DEVICES\_XML=$(DEVICES\_XML) \**  **perl $(MTK\_PTGEN\_PERL)**    **ifneq ($(CALLED\_FROM\_SETUP),true)**  **diff --git a/build/tools/ptgen/MT6765/ptgen.pl b/build/tools/ptgen/MT6765/ptgen.pl**  **index 84f88fb..7cc9d0a 100755**  **--- a/build/tools/ptgen/MT6765/ptgen.pl**  **+++ b/build/tools/ptgen/MT6765/ptgen.pl**  **@@ -5,6 +5,7 @@**  **use strict;**  **use File::Basename;**  **use File::Path;**  **+use XML::Simple;**  **my $LOCAL\_PATH;**    **BEGIN**  **@@ -1001,6 +1002,20 @@ sub GenYAMLScatFile()**  **$Scatter\_Info{$part->{Partition\_Name}}{is\_download} = "true";**  **}**    **+ if ($part->{Partition\_Name} eq "md1img" || $part->{Partition\_Name} eq "oem")**  **+ {**  **+ $Scatter\_Info{$part->{Partition\_Name}}{is\_oem} = "true";**  **+ }**  **+ else**  **+ {**  **+ $Scatter\_Info{$part->{Partition\_Name}}{is\_oem} = "false";**  **+ }**  **+**  **+ if ($part->{Rename\_Prefix})**  **+ {**  **+ $Scatter\_Info{$part->{Partition\_Name}}{rename\_prefix} = $part->{Rename\_Prefix};**  **+ }**  **+**  **if ($part->{OTA\_Update} eq "N")**  **{**  **$Scatter\_Info{$part->{Partition\_Name}}{is\_upgradable} = "false";**  **@@ -1048,6 +1063,14 @@ \_\_TEMPLATE**  **##########################################################################################**  **\_\_TEMPLATE**    **+ my $Head3 = <<"\_\_TEMPLATE";**  **+##########################################################################################**  **+#**  **+# Variant Infos**  **+#**  **+##########################################################################################**  **+\_\_TEMPLATE**  **+**  **my ${FirstDashes} = "- ";**  **my ${FirstSpaceSymbol} = " ";**  **my ${SecondSpaceSymbol} = " ";**  **@@ -1085,6 +1108,11 @@ \_\_TEMPLATE**  **print $ScatterFileFH "${FirstSpaceSymbol}partition\_name${colon}$part->{Partition\_Name}\n";**  **print $ScatterFileFH "${FirstSpaceSymbol}file\_name${colon}$Scatter\_Info{$part->{Partition\_Name}}{file\_name}\n";**  **print $ScatterFileFH "${FirstSpaceSymbol}is\_download${colon}$Scatter\_Info{$part->{Partition\_Name}}{is\_download}\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}is\_oem${colon}$Scatter\_Info{$part->{Partition\_Name}}{is\_oem}\n";**  **+ if ($Scatter\_Info{$part->{Partition\_Name}}{rename\_prefix})**  **+ {**  **+ print $ScatterFileFH "${FirstSpaceSymbol}rename\_prefix${colon}$Scatter\_Info{$part->{Partition\_Name}}{rename\_prefix}\n";**  **+ }**  **print $ScatterFileFH "${FirstSpaceSymbol}type${colon}$Scatter\_Info{$part->{Partition\_Name}}{type}\n";**  **print $ScatterFileFH "${FirstSpaceSymbol}linear\_start\_addr${colon}$Scatter\_Info{$part->{Partition\_Name}}{linear\_start\_addr}\n";**  **print $ScatterFileFH "${FirstSpaceSymbol}physical\_start\_addr${colon}$Scatter\_Info{$part->{Partition\_Name}}{physical\_start\_addr}\n";**  **@@ -1098,6 +1126,25 @@ \_\_TEMPLATE**  **print $ScatterFileFH "${FirstSpaceSymbol}empty\_boot\_needed${colon}$Scatter\_Info{$part->{Partition\_Name}}{empty\_boot\_needed}\n";**  **print $ScatterFileFH "${FirstSpaceSymbol}reserve${colon}0x00\n\n";**  **}**  **+ if (-e $ENV{DEVICES\_XML})**  **+ {**  **+ print $ScatterFileFH $Head3;**  **+ my $Device\_XML = XMLin($ENV{DEVICES\_XML});**  **+ my %Device\_Info = %{$Device\_XML->{device}};**  **+ my $device\_index= 0;**  **+ foreach my $key (keys(%Device\_Info))**  **+ {**  **+ print $ScatterFileFH "${FirstDashes}device\_index${colon}$device\_index\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}name${colon}$key\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}product${colon}$Device\_Info{$key}{'product'}\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}model${colon}$Device\_Info{$key}{'model'}\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}modem\_tag${colon}$Device\_Info{$key}{'modem\_tag'}\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}oem\_tag${colon}$Device\_Info{$key}{'oem\_tag'}\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}svnnumber${colon}$Device\_Info{$key}{'svnnumber'}\n";**  **+ print $ScatterFileFH "${FirstSpaceSymbol}oem\_code${colon}$Device\_Info{$key}{'oem\_code'}\n\n";**  **+ $device\_index++;**  **+ }**  **+ }**  **close $ScatterFileFH;**  **}** |

# build property生成

原生几个ro.product属性从在MakeFile中设置好，并通过shell脚本写入到build.prop文件中，因此，根据几个ro.boot属性设置时，需要先将buildinfo.sh中ro.product.model, ro.product.brand, ro.product.name, ro.product.device, ro.build.product通过宏控制起来，只在不启用FCM方案时有效。

上述5个system property,加上publicname和ecid会在init.cpp中根据ro.boot属性设置。

|  |
| --- |
| init.cpp:  static void export\_kernel\_boot\_props() {  struct {  const char \*src\_prop;  const char \*dst\_prop;  const char \*default\_value;  } prop\_map[] = {  { "ro.boot.serialno", "ro.serialno", "", },  { "ro.boot.mode", "ro.bootmode", "unknown", },  { "ro.boot.baseband", "ro.baseband", "unknown", },  { "ro.boot.bootloader", "ro.bootloader", "unknown", },  { "ro.boot.hardware", "ro.hardware", "unknown", },  { "ro.boot.revision", "ro.revision", "0", },  { "ro.boot.ecid", "ro.ecid", "00000001", },  { "ro.boot.brand", "ro.product.brand ", "unknown", },  { "ro.boot.model", "ro.product.model", " unknown ", },  { "ro.boot.product ", "ro.build.product", " unknown ", },  { "ro.boot.name ", "ro.product.name ", " unknown ", },  { "ro.boot.publicname", "ro.product.publicname ", " unknown ", },  { "ro.boot.product", "ro.product.device ", " unknown ", },  };  for (size\_t i = 0; i < arraysize(prop\_map); i++) {  std::string value = GetProperty(prop\_map[i].src\_prop, "");  property\_set(prop\_map[i].dst\_prop, (!value.empty()) ? value : prop\_map[i].default\_value);  }  } |

Vendor下同样有ro.product.vendor.brand等属性，也同样要先在vendor\_buildinfo.sh中先将这几个vendor的product属性控制住，当启动FCM方案时，不再向vendor的build.prop中写入。我们会在vendor/ect/init/下新加一个init.vendor.chome.rc，此处会设置vendor的product属性。

|  |
| --- |
| **init.vendor.chome.rc:**  on post-fs  setprop ro.product.vendor.brand ${ro.boot.brand}  setprop ro.product.vendor.device ${ro.boot.product}  setprop ro.product.vendor.model ${ro.boot.model}  setprop ro.product.vendor.name ${ro.boot.name} |

# fingerprint生成

**system fingerprint 和bootimage fingerprint处理。**

**fingerprint会在property\_service.cpp#load\_system\_props中去设置。**

**原生fingerprint的拼接规则为:**

**BUILD\_FINGERPRINT := $(PRODUCT\_BRAND)/$(TARGET\_PRODUCT)/$(TARGET\_DEVICE):$(PLATFORM\_VERSION)/$(BUILD\_ID)/$(BF\_BUILD\_NUMBER):$(TARGET\_BUILD\_VARIANT)/$(BUILD\_VERSION\_TAGS)**

**其中PRODUCT\_BRAND，TARGET\_PRODUCT，TARGET\_DEVICE分别对应ro.product.brand, ro.product.name, ro.product.device。这三个已经根据ro.boot属性设置好。PLATFORM\_VERSION对应ro.boot.version.release。BUILD\_ID对应ro.build.id。BF\_BUILD\_NUMBER会在设置完fingerprint后清空，需要在MakeFile中重新添加一个VND\_BUILD\_NUMBER,将其赋予ro.build.tnumber和ro.vendor.build.tnumber分别写入到system和vendor下的build.prop。TARGET\_BUILD\_VARIANT对应ro.build.type。BUILD\_VERSION\_TAGS对应ro.build.tags。拼接完成之后将ro.build.fingerprint和ro.bootimagebuild.fingerprint设置成拼接成的fingerprint。**

|  |  |  |
| --- | --- | --- |
| **原生fingerprint组成部分** | **FCM使用的system property** | **来源** |
| **PRODUCT\_BRAND** | **ro.product.brand** | **ro.boot.brand** |
| **TARGET\_PRODUCT** | **ro.product.name** | **ro.boot.name** |
| **TARGET\_DEVICE** | **ro.product.device** | **ro.boot.product** |
| **PLATFORM\_VERSION** | **ro.boot.version.release** | **原生，编译时来自PLATFORM\_VERSION** |
| **BUILD\_ID** | **ro.build.id** | **原生，编译时来自BUILD\_ID** |
| **BF\_BUILD\_NUMBER** | **ro.build.tnumbe** | **将BF\_BUILD\_NUMBER保存到ro.build.tnumbe** |
| **TARGET\_BUILD\_VARIANT** | **ro.build.type** | **原生，编译时来自TARGET\_BUILD\_VARIANT** |
| **BUILD\_VERSION\_TAGS** | **ro.build.tags** | **原生，编译时来自BUILD\_VERSION\_TAGS** |

**Vendor分区下面也同样需要一个vendor fingerpint。由于vts测试会刷system分区，不能在system分区处理或读取system分区下配置的system property。我们会在vendor/ect/init/下新加一个init.vendor.chome.rc，此处设置vendor 的fingerprint。**

**Setprop ro.vendor.build.fingerprint ${ro.boot.brand:-unknown}/${ro.boot.name:-unknown}/${ro.boot.product:-unknown}:${ro.vendor.version.release:-unknown}/${ro.vendor.build.id:-unknown}/${ro.vendor.build.tnumber:-unknown}:${ro.vendor.build.type:-unknown}/${ro.vendor.build.tags:-unknown}**

**其中，像type之类的属性是保存在system中的，因此需要在vendor下设置同样的属性供vendor fingerprint使用。ro.build.type对应vendor下ro.vendor.build.type，写入vendor下的build.prop中。**

# 根据ecid配置system property

system prop支持根据根据ecid配置，将build.prop文件放在FCM对应的配置目录下（/system/aio\_custim/resources/common\_[ecid]/build.prop）在property\_service.cpp的load\_system\_props会读取system,vendor, odm等build.prop中的配置，最后添加根据ecid读取对应目录下prop文件的配置。

|  |
| --- |
| void load\_system\_props() {  load\_properties\_from\_file("/system/build.prop", NULL);  load\_properties\_from\_file("/odm/build.prop", NULL);  load\_properties\_from\_file("/vendor/build.prop", NULL);  load\_properties\_from\_file("/factory/factory.prop", "ro.\*");  load\_recovery\_id\_prop();  std::string ecid = android::base::GetProperty("ro.ecid", "00000001");  std::string path = StringPrintf("/system/aio\_custom/resources/common\_%s/build.prop", ecid.c\_str());  load\_properties\_from\_file(path.c\_str(), null);  } |