



Unisoc Confidential For hiar

时区开发客制化指导手册

文档版本
发布日期

V1.1
2020-06-04

版权所有 © 紫光展锐科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

Unisoc Confidential For hiar

紫光展锐科技有限公司



前言

概述

本文档详细地描述了时区升级、时区设置及常见问题的解决方法。

读者对象


本文档主要适用于修改时区相关问题的软件开发人员。

缩略语

缩略语	英文全名	中文解释
Tzdata	Time Zone Database	时区信息数据库
IANA	Internet Assigned Numbers Authority	互联网数字分配机构
ICU	International Component for Unicode	Unicode 国际化组件
CTS	Compatibility Test Suite	兼容性测试包

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.0	2020-02-20	第一次正式发布。
V1.1	2020-06-04	1. 调整文档章节结构，优化文档描述

文档版本	发布日期	修改说明
		<ul style="list-style-type: none">2. 添加 Android Q 平台相关描述3. 添加默认时区设置步骤及示例4. 文档名称由《时区开发客制化文档》修改成《时区开发客制化指导手册》

关键字

时区、Tzdata、ICU。

Unisoc Confidential For hiar

目 录

1 概述	1
1.1 Tzdata 简介	1
1.2 ICU 简介	1
2 Tzdata 升级	2
2.1 背景	2
2.2 升级方法	2
2.2.1 Android4.4	3
2.2.2 Android8.1	5
2.2.3 Android9.0	6
2.2.4 AndroidQ	6
2.3 注意事项	6
3 ICU 编译	8
3.1 背景	8
3.2 编译方法	8
3.2.1 Android4.4	8
3.2.2 Android8.1	9
3.2.3 Android9.0	9
3.2.4 AndroidQ	9
3.3 注意事项	9
4 时区开发指导	11
4.1 默认时区配置	11
4.1.1 配置方法	11
4.1.2 调试指导	12
4.2 时区显示名称变更	14
4.2.1 背景	14
4.2.2 操作方法	14
4.3 自定义时区名称显示	15
4.3.1 背景	15
4.3.2 操作方法	16
4.4 时区添加	17
4.4.1 背景	17
4.4.2 操作方法	17

图目录

图 2-1 Tzdata 文件版本信息	2
图 4-1 Android8.1 与 Android Q 莫斯科时区数据对比	15

Unisoc Confidential For hiar

1 概述

时区数据由两个重要部分组成：**Tzdata**（Time Zone Database）和 **ICU**（International Component for Unicode）。其中 **Tzdata** 定义了时区规则，**ICU** 包含时区显示规则。

当时区规则有更新时，需要升级 **Tzdata** 以获取最新的时区规则；而时区显示有变更时，则需要重新编译 **ICU** 以获取最新的时区显示。

1.1 Tzdata 简介

Tzdata 是时区信息数据库（也称 **Olson database**），由 **IANA**（Internet Assigned Numbers Authority）统一维护。

Tzdata 源数据文件定义了世界上的时区和城市信息、夏令时等一些时间转换信息，以文本文件的形式发布，并以一定的格式将时区信息记录下来。如记录一个城市的时区 ID（如“Asia/Shanghai”）、某地区夏令时的 **Rule** 等。

1.2 ICU 简介

ICU（International Component for Unicode）是 **Unicode** 国际化组件的缩写，用于支持软件国际化的开源项目。

ICU4C 是 **ICU** 在 **C/C++** 平台下的版本。**ICU4C** 提供了 **C/C++** 平台强大的国际化开发能力，根据各地风俗和语言习惯，实现对数字、货币、时间、日期和消息的格式化、解析，对字符串进行大小写转换、整理、搜索和排序等功能。

更多功能可以通过 **ICU** 官方链接了解：<http://site.icu-project.org/home>。

2 Tzdata 升级

2.1 背景

由于一些国家和地区每年可能会对时区政策进行调整，如启用或取消夏令时，tzdata 文件每年也会不时地进行更新，因此产生了各个版本的 Tzdata 文件，如 tzdata2018e、tzdata2018i 等。

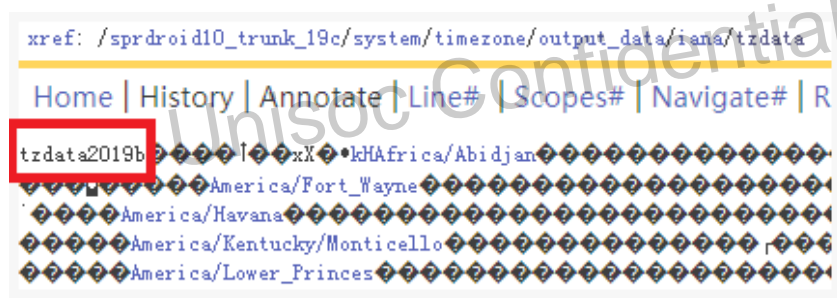
同一个城市时区 ID 在不同版本的 Tzdata 中定义的时区规则可能有所不同，也有可能某个版本的 Tzdata 中被取消了，最新版本的 Tzdata 源数据可以从 IANA 的官方链接上获取：

- <http://www.iana.org/timezones>
- <ftp://ftp.iana.org/tz/releases>

Android 手机上的时区信息是保存在 Tzdata 二进制文件中，是 Google 根据 Tzdata 源数据执行脚本生成的，由 Google 维护。

Tzdata 文件的开头包含版本信息，详见图 2-1。

图2-1 Tzdata 文件版本信息



2.2 升级方法

由于城市的时区规则可能会发生变化，Tzdata 的版本也一直在更新，低版本的 Android 设备如果需要正确显示这些城市的时间，就需要对 Tzdata 进行升级。

Android 版本不同，Tzdata 的升级过程也有所差异。

说明

文档代码描述中“+”代表需要新增的代码，“-”表示需要删除的代码，请勿直接粘贴代码进行编译。

2.2.1 Android4.4

Android4.4 平台升级 Tzdata 的脚本工具路径为：bionic/libc/tools/zoneinfo/update-tzdata.py，该工具中包含下载新的 Tzdata 源数据、验证下载的源数据、解析并生成新的 Tzdata 等功能。

此脚本工具执行一次只会升级一个版本，升级到最新版本，需要多次执行。如需直接升级到最新版本，需要修改脚本工具。

下面以升级到目前最新的版本 2019c 作为示例介绍修改方法，步骤如下：

- 步骤 1 从 [ftp://ftp.iana.org/tz/releases](http://ftp.iana.org/tz/releases) 获取最新的 Tzdata 源数据 tzdata2019c.tar.gz，并将该源数据放置到脚本工具所在的 bionic/libc/tools/zoneinfo 目录下。
- 步骤 2 update-tzdata.py 脚本，去除脚本工具文件中网络下载源数据、验证源数据签名等功能，指定源数据包，解析并生成新的 Tzdata，具体如下：

```
--- a/libc/tools/zoneinfo/update-tzdata.py
+++ b/libc/tools/zoneinfo/update-tzdata.py
@@ -10,6 +10,7 @@ import subprocess
import sys
import tarfile
import tempfile
+import shutil
# Find the bionic directory, searching upward from this script.
bionic_libc_tools_zoneinfo_dir = os.path.realpath(os.path.dirname(sys.argv[0]))
@@ -109,19 +110,19 @@ def HttpUpgrade(http, data_filename):
def ExtractAndCompile(data_filename):
    new_version = re.search('(tzdata.+)\.tar\.gz', data_filename).group(1)

    signature_filename = '%s.asc' % data_filename
    print 'Verifying signature...'
    # If this fails for you, you probably need to import Paul Eggert's public key:
    # gpg --recv-keys ED97E90E62AA7E34
    subprocess.check_call(['gpg', '--trusted-key=ED97E90E62AA7E34', '--verify',
                           signature_filename, data_filename])

    print 'Extracting...'
+   if os.path.exists('extracted'):
+       shutil.rmtree('extracted')
+
    os.mkdir('extracted')
    tar = tarfile.open(data_filename, 'r')
    tar.extractall('extracted')
    print 'Calling zic(1)...'
+   if os.path.exists('data'):
+       shutil.rmtree('data')
+

```

```
tzdata_filenames.append(filename)
data_filenames.sort()
e:
tp = urllib.HTTPConnection('www.iana.org')
tp.request("GET", "/time-zones")
data_lines = http.getresponse().read().split('\n')
```

```
- HttpUpgrade(http, filename)
- sys.exit(0)
+ ExtractAndCompile(data_filename)

print 'You already have the latest Tzdata (%s)!' % current_version
sys.exit(0)
```

步骤3 执行命令：`python update-tzdata.py` 进行升级操作。

命令执行完成后，`bionic/libc/zoneinfo/tzdata` 会更新为新的版本。

----结束

2.2.2 Android8.1

Android8.1 升级 Tzdata 的脚本工具包括两个脚本文件：`system/timezone/download-iana-data.py` 和 `system/timezone/update-tzdata.py`。

- `download-iana-data.py`: 用于从网络下载最新的 Tzdata 源数据
- `update-tzdata.py`: 用于解析源数据生成最新的 Tzdata

Android 源码中提供的生成 Tzdata 的脚本 `update-tzdata.py`，其中包含一些校验功能，所以执行 `update-tzdata.py` 脚本时会出现错误，需要根据具体情况修改脚本后才能执行。

下述升级步骤中详细描述了脚本修改方法。

步骤1 全编译 Android 源码。

步骤2 修改 `download-iana-data.py` 脚本。

```
diff --git a/download-iana-data.py b/download-iana-data.py
index 3792806..afd3338 100755
--- a/download-iana-data.py
+++ b/download-iana-data.py
@@ -46,8 +46,8 @@ def FtpRetrieveFileAndCheckSignature(ftp, data_filename):
    print 'Verifying signature...'
    # If this fails for you, you probably need to import Paul Eggert's public key:
    # gpg --recv-keys ED97E90E62AA7E34
-    subprocess.check_call(['gpg', '--trusted-key=ED97E90E62AA7E34', '--verify',
-                            signature_filename, data_filename])
+    #subprocess.check_call(['gpg', '--trusted-key=ED97E90E62AA7E34', '--verify',
+    #                        signature_filename, data_filename])
```

步骤3 执行命令：`python download-iana-data.py` 下载最新的 Tzdata 源数据。

步骤4 修改 `update-tzdata.py` 脚本。

```
diff --git a/update-tzdata.py b/update-tzdata.py
index 20986c7..1b1ff97 100755
--- a/update-tzdata.py
```

```
+++ b/update-tzdata.py
@@ -228,7 +228,7 @@ def main():
    icu_dir = icuutil.icuDir()
    print 'Found icu in %s ...' % icu_dir

-   BuildIcuData(iana_data_tar_file)
+   #BuildIcuData(iana_data_tar_file)
```

步骤 5 执行命令 `python update-tzdata.py` 生成新的 tzdata。

此时最新的 tzdata 已经生成，可以进入 `system/timezone/output_data/iana/` 目录下查看 tzdata 的版本号为最新的 2019c。

步骤 6 全编译工程并验证。

此时新的 tzdata 会生效，可以直接将 tzdata 文件 push 到手机中的 `/system/usr/share/zoneinfo` 目录进行验证。

---结束

2.2.3 Android9.0

Android9.0 平台除 `download-iana-data.py` 脚本修改方法与 Android8.1 平台略有差异外，其他步骤完全同 Android8.1，Android8.1 平台 tzdata 升级请参见“2.2.2 Android8.1”。

Android9.0 平台 `download-iana-data.py` 脚本修改方法如下：

```
diff --git a/download-iana-data.py b/download-iana-data.py
index 6b9865b..81d6fe5 100755
--- a/download-iana-data.py
+++ b/download-iana-data.py
@@ -90,7 +90,7 @@ def main():
    print 'Downloading signature (%s)...' % signature_filename
    FtpRetrieveFile(ftp, signature_filename)

-   CheckSignature(latest_iana_tar_filename, signature_filename)
+   # CheckSignature(latest_iana_tar_filename, signature_filename)
```

2.2.4 AndroidQ

由于 AndroidQ 平台 Tzdata 已经 mainline，由 Google 统一进行推送，Android 代码中也已经删除了升级相关的工具，用户可以通过 Google 应用市场升级 Tzdata，此处不再赘述。

2.3 注意事项

升级 Tzdata 通常可能会导致 CTS（Compatibility Test Suite）测试失败，因为 Google 并不是一直在维护每个 Android 大版本的 CTS 测试包的，例如目前 Google 还在维护的只有 Android8.1、Android9.0、AndroidQ；而 Android 8.1 之前的版本 CTS 测试包已经停止更新。

出现 CTS 测试失败时，原因及解决方法如下：

- **Android8.1 之前版本：**当 Tzdata 升级之后，测试包的期望值是旧的错误的值，而 Tzdata 更新后测试结果返回的是新的正确的值，此时就会出现不匹配的情况，CTS 测试会失败。
 - 如果不需要过 CTS 认证，则不需要关注。
 - 如果要过 CTS，则需要提 Google issue 与 Google 沟通确认。
- **Android8.1 & Android9.0 版本：**因为 CTS 测试包还在更新，所以当 CTS 测试失败时可以获取最新正式发布的测试包进行测试。

如果测试失败，可以尝试获取 Google dailybuild 版本的 CTS 测试包进行验证；如果仍然不过，需要提 Google issue 与 Google 进行沟通确认。
- **AndroidQ：**AndroidQ 平台 Tzdata 已经 mainline，不存在 CTS 测试失败的问题。

Unisoc Confidential For hiar

3 ICU 编译

3.1 背景

Android 版本中集成了 ICU 组件，Android8.1 和 Android9.0 的代码是使用 ICU4J 实现的，而数据则使用了 ICU4C 中的数据。

当客户有特定显示需求或者时区数据显示有变更时，需要重新编译 ICU 生成 dat 文件。

3.2 编译方法

不同的平台版本，ICU 的编译方式不同。

3.2.1 Android4.4

如果只是测试修改后效果，执行步骤 1 即可；如果需要 build 后生效，则需要执行全部步骤。

Android4.4 平台 ICU 编译步骤如下：

步骤 1 在临时目录中编译 ICU 资源。

- a. 在 external/icu4c 下新建临时目录 icubuild。
`$mkdir external/icu4c/icuBuild`
- b. 进入 icubuild 目录。
`$cd external/icu4c/icuBuild`
- c. 执行 icuConfigureRun Linux 命令，生成 make 文件。
`$./runConfigureICU Linux`
- d. 执行 make -j2 命令。
`$make -j2`
- e. 将生成的 external/icu4c/icuBuild/data/out/tmp/icudt51l.dat push 到手机测试。
`$adb root`
`$adb remount`
`$adb push external/icu4c/icuBuild/out/data/tmp/icudt51l.dat system/usr/icu/`
- f. 重启手机，查看修改后的效果。
`$adb reboot`

步骤 2 将生成的 external/icu4c/icuBuild/data/out/tmp/icudt51l.dat 文件复制到 external/icu4c/stubdata 下并改名为 icudt51l-all.dat，覆盖原来的同名文件。

步骤 3 配置编译环境。

- a. 进入代码根目录，执行命令配置 jdk 版本。

```
$source /usr/local/bin/change_to_v4.sh
```

- b. 配置环境变量。

```
$source build/envsetup.sh
```

```
$lunch
```

```
$kheader
```

步骤 4 执行脚本，重新生成.dat 文件。

- a. 进入 external/icu4c/stubdata 目录

```
$cd external/icu4c/stubdata
```

- b. 执行脚本

```
$/icu_dat_generator.py
```

步骤 5 重新编译工程并验证。

```
$make
```

---结束

3.2.2 Android8.1

Android8.1 & Android9.0 原生平台提供了编译 ICU 的脚本，具体为 external/icu/icu4c/source/makeData.sh。

Android8.1 & Android9.0 平台 ICU 编译步骤如下：

步骤 1 执行命令 ./makeData.sh。

命令执行完成后，Android8.1 会生成 external/icu/icu4c/source/stubdata/icudt58l.dat。

步骤 2 重新编译 Android 工程并验证。

---结束

3.2.3 Android9.0

Android9.0 平台编译步骤同 Android8.1 平台，编译方法请参见“3.2.2 Android8.1”。

Android9.0 平台执行完 ./makeData.sh 命令后会生成 external/icu/icu4c/source/stubdata/icudt60l.dat。

3.2.4 AndroidQ

由于 Google 关闭了 AndroidQ 平台 ICU 编译功能，修改了 ICU 文件也无法编译验证，此处不再赘述 AndroidQ 平台相关更改。

3.3 注意事项

ICU 更新的注意事项与 Tzdata 升级的注意事项相同，主要是可能导致 CTS 测试失败。

如果项目需要过 CTS 认证，更新 ICU 后，需要进行 CTS 测试；如果测试失败，需要跟 Google 沟通确认。

Unisoc Confidential For hiar

4 时区开发指导

时区开发主要包含：设置默认时区、更新时区显示名称、自定义时区名称及新增时区。

不同平台时区相关信息存储在不同的文件中：

- Android4.4 平台中设置中的时区显示列表是在 `package/apps/Settings/res/xml-xx/timezones.xml` 中定义的，而系统支持的时区 ID 是在 `frameworks/base/core/res/res/xml/time_zones_by_country.xml` 文件中定义的。
其中 `xml-xx` 中的 `xx` 表示的是时区显示语言类型，如中文显示，则是 `xml-zh`。
- Android8.0 以及之后的版本，设置中的时区列表是由系统中的 `system/timezone/output_data/android/tzlookup.xml` 文件定义的，该文件就是当前设备支持的所有时区列表，只要在 `tzlookup.xml` 文件有定义的时区 id，都会在 Setting 中显示。

说明

文件中涉及代码的描述“+”代表需要新增的代码，“-”表示需要删除的代码，请勿直接粘贴代码进行编译。

4.1 默认时区配置

4.1.1 配置方法

不同平台默认时区设置方法有所差异。

4.1.1.1 Android4.4

Android4.4 平台默认时区设置方法如下：

步骤 1 查询要设置为默认时区的时区 ID 是否支持。

即查看需要设置默认时区的时区 ID 是否在 `frameworks/base/core/res/res/xml/time_zones_by_country.xml` 文件中有定义。

- 如有，执行步骤 2。
- 如无，需要先添加该时区，然后执行步骤 3。时区添加方法请参见“4.4 时区添加”。

步骤 2 查看 `package/apps/Settings/res/xml-xx/timezones.xml` 中是否有该默认时区的定义。

- 如有，执行步骤 3。
- 如无，需要将默认时区的 ID 添加到 `timezones.xml` 后才能执行步骤 3。

以添加上海时区为默认时区为例，添加方法如下：

```
<timezones>
  <timezone id="Pacific/Fiji">Fiji</timezone>
  <timezone id="Pacific/Tongatapu">Tonga</timezone>
```

```
+ <timezone id="Asia/Shanghai">Shanghai</timezone>
</timezones>
```

步骤 3 配置系统属性 `persist.sys.timezone` 的值为默认时区的 ID。

该系统属性需要在 `device/sprd` 目录下产品对应 board 的 `system.prop` 文件中添加，如设置上海为默认时区，配置为：`persist.sys.timezone=Asia/Shanghai`。

步骤 4 编译版本并验证。

```
$make
```

---结束

4.1.1.2 Android8.1

Android8.1 & Android9.0 & AndroidQ 平台默认时区设置方法如下：

步骤 1 查询要设置为默认时区的时区 ID 是否支持。

即查看需要设置默认时区的时区 ID 是否在 `tzlookup.xml` 文件中有定义。

- 如有，执行步骤 2。
- 如无，需要先添加该时区，然后执行步骤 2。时区添加方法请参见“4.4 时区添加”。

步骤 2 配置系统属性 `persist.sys.timezone` 的值为默认时区的 ID。

该系统属性需要在 `device/sprd` 目录下产品对应 board 的 `system.prop` 文件中添加，如设置上海为默认时区，配置为：`persist.sys.timezone=Asia/Shanghai`。

步骤 3 编译版本并验证。

```
$make
```

---结束

4.1.1.3 Android9.0

Android9.0 平台默认时区设置同 Android8.1，具体请参见“4.1.1.2 Android8.1”。

4.1.1.4 AndroidQ

AndroidQ 平台默认时区设置同 Android8.1，具体请参见“4.1.1.2 Android8.1”。

4.1.2 调试指导

配置完默认时区后会有不生效的情况产生，可能的原因及解决方法如下：

- **原因 1：**自动更新时区默认开启，插卡开机，有网络连接，开机后自动更新了时区，导致默认时区不生效。
解决方法：自动更新时区开关默认关闭，即将自动更新时区开关 `def_auto_time_zone` 的值设置为 `false`。文件路径：`platform/frameworks/base/packages/SettingsProvider/res/values/defaults.xml`
- **原因 2：**自动更新时区默认开启，不插卡开机，不连接网络，开机后自动更新了时区，导致默认时区不生效。

解决方法：将自动更新时区开关 `def_auto_time_zone` 的值设置为 `false`。

如果客户会希望保持自动更新时区开关打开，在不插 `sim` 卡的情况下开机不更新时区，可通过如下方法进行修改：

- Android4.4

`frameworks/opt/telephony/src/java/com/android/internal/telephony/gsm/GsmServiceStateTracker.java` 文件的 `setAndBroadcastNetworkSetTimeZone` 方法开头添加：

```
private void setAndBroadcastNetworkSetTimeZone(String zoneId) {
+    // Determine if the Icc card exists
+    boolean iccCardExist = false;
+    if (mUiccApplication != null) {
+        iccCardExist = mUiccApplication.getState() !=
AppState.APPSTATE_UNKNOWN;
+    }
+    if (!iccCardExist) {
+        return;
+    }
+    ..... //此处代码省略
}
```

- Android 8.1

`frameworks/opt/telephony/src/java/com/android/internal/telephony/ServiceStateTracker.java` 文件的 `setAndBroadcastNetworkSetTimeZone` 方法开头添加：

```
private void setAndBroadcastNetworkSetTimeZone(String zoneId) {
+    // Determine if the Icc card exists
+    boolean iccCardExist = false;
+    if (mUiccApplication != null) {
+        iccCardExist = mUiccApplication.getState() !=
AppState.APPSTATE_UNKNOWN;
+    }
+    if (!iccCardExist) {
+        return;
+    }
+    ..... //此处代码省略
}
```

- Android9.0 & AndroidQ

`frameworks/opt/telephony/src/java/com/android/internal/telephony/NewNitzStateMachine.java` 文件的 `setAndBroadcastNetworkSetTimeZone` 方法开头添加如下代码，对应的类也需要导入：

```
+ import com.android.internal.telephony.uicc.IccCardApplicationStatus.AppState;
+ import com.android.internal.telephony.uicc.UiccCardApplication;
private void setAndBroadcastNetworkSetTimeZone(String zoneId) {
+    UiccCardApplication uiccApp = mPhone.getUiccCardApplication();
+    if (uiccApp == null || uiccApp.getState() == AppState.APPSTATE_UNKNOWN) {
+        Rlog.d(LOG_TAG, "Not to set network time zone due to sim absent.");
+    }
```

```
+     return;
+ }
+
+     ..... //此处代码省略
+ }
```

- **原因 3:** 设置的默认时区不在支持的时区列表中，显示了其他时区，导致默认时区不生效。

如果是默认时区列表文件中没有的时区 id，配置为默认时区是无效的，因为系统不支持。

解决方法: 增加需要默认显示的时区，操作方法请参考“4.4 时区添加”。

- **原因 4:** 系统带有 Google 开机向导，Google 开机向导中有时区设置界面，如果设置的默认时区在其列表中不存在，开机向导会显示其他时区，这时如果点击下一步，会将开机向导界面显示的时区设置到系统中，导致默认时区不生效。

解决方法: 无。Google 开机向导应用维护的时区列表是独立的，与系统支持的时区列表可能不一致；而且 Google 开机向导应用是闭源的，所以无法修改。

- **原因 5:** selinux 权限异常，导致默认时区不生效，通常见于 Android8.1 & Android9.0 & AndroidQ 平台。

如果是因为权限问题不生效，kernel log 中会有如下信息打印，可以作为参考：

```
.....selinux: avc: denied { set } for property=persist.sys.timezone pid=1 uid=0 gid=0
scontext=u:r:vendor_init:s0 tcontext=u:object_r:exported_system_prop:s0 tclass=property service
permissive=0
.....init: Unable to set property 'persist.sys.timezone' to 'Europe/Amsterdam' in property file
'/vendor/build.prop': SELinux permission check failed
```

解决方法: 添加相应的权限即可，具体修改如下：

```
device/sprd/XXX/common/sepolicy/vendor_init.te
+ allow vendor_init exported_system_prop:property_service { set }
```

4.2 时区显示名称变更

4.2.1 背景

目前一些国家有时会颁布法令更新时区的规则。例如俄罗斯莫斯科地区时区由 GMT + 4 调整为 GMT + 3；卡萨布兰卡从 2018 年 10 月 28 日之后，将标准时从 GMT + 0 调整为 GMT + 1，夏令时从 + 1 小时变更为 - 1 小时等。此时如果想要显示正式的时区信息，就需要更新时区数据库以及 ICU 资源。

4.2.2 操作方法

以俄罗斯时区规则修改为例，操作方法如下：

步骤 1 更新时区规则 Tzdata

Tzdata 的更新过程请参考“2.2 升级方法”。

步骤 2 更新 ICU 资源

ICU 资源涉及时区的最终显示，目前 ICU 中与时区更新有关的文件有很多，其中涉及到规则的有如下 4 个，这些文件是 Google 通过执行脚本生成的。

- icu4c/source/data/misc/metaZones.txt

4.3.2 操作方法

以荷兰阿姆斯特丹为例，如果要修改荷兰语下显示为“阿姆斯特丹时间”或其他自定义的显示，请按照如下步骤进行修改：

步骤 1 查找时区 id

时区 id 并不是城市名称，手机显示到时区列表上的时区名称与时区 id 有映射关系。

- Android4.4 平台可以通过查看 Settings 中的 timezones.xml 来确定，package/apps/Settings/res/xml-zh/timezones.xml 是中文显示的时区文件，文件中 Amsterdam 的时区 id 是 Europe/Amsterdam。
- Android8.1 和 Android9.0 平台可以通过查看 system/timezone/output_data/android/tzlookup.xml 来确定。

步骤 2 修改 metaZones.txt 文件

metaZones.txt 文件中需要修改两个类文件：mapTimezones、metaZoneinfo。

- 修改类 mapTimezones

修改的目的是把时区指向自定义的显示规则中，这样通过时区 ID 就可以得到自定义的显示规则，并显示到界面上。

- 将自定义的 Europe_Amsterdam 新规则添加到 mapTimezones 中，mapTimezones 中规则是按字母排序的，所以 Europe_Amsterdam 这个规则要添加到 Europe_Central 的前面，添加的新规则如下：

```
Europe_Amsterdam{
}

```

- 将要修改的时区从原规则中删除，添加到自定义的规则中。

找到要修改的时区，例如 Europe/Amsterdam 在 Europe_Central 中：

剪切 NL{“Europe/Amsterdam”}，粘贴到自定义的时区显示规则中，如下所示：

```
Europe_Amsterdam{
    NL{"Europe/Amsterdam"}
}

```

- 修改类 metaZoneinfo

添加 meta 信息,这样就可以通过显示规则名称 Europe_Amsterdam 得到 Europe:Amsterdam 的 meta 信息。

```
"Europe:Amsterdam"{
    {
        "Europe_Amsterdam"
    }
}

```

步骤 3 修改 zone/nl.txt 文件

不同的 txt 文件代表不同语言，语言和区域代码可通过查询 ISO-3166-1 和 ISO-639 标准确定。

zone 目录下的文件定义了界面显示的字符串，根据上一步骤中得到的 meta 信息可以得到要显示的字符串。

同样的，名称是按照字母排序的，所以在 Europe_Central 前面添加，如下：

```
"meta: Europe_Amsterdam" {
    ld{ "夏令时间显示名称" }
    ls{ "标准时间显示名称" }
}
```

步骤 4 编译 ICU 资源并验证

请参见“3.2 编译方法”。

---结束

4.4 时区添加

4.4.1 背景

目前一些国家存在多个时区或多个地区共用一个时区，或一个国家没有时区，遇到这种情况，客户有时会要求新增时区。

4.4.2 操作方法

不同平台操作步骤也不尽相同，下面按照平台来介绍新增时区操作方法。

4.4.2.1 Android4.4

设置中的时区显示列表是在 package/apps/Settings/res/xml-xx/timezones.xml 中定义的，而系统支持的时区 ID 是在 frameworks/base/core/res/res/xml/time_zones_by_country.xml 文件中定义的。

新增时区时首先要查询要添加的时区 ID 是否支持，即查看添加的时区 ID 在文件 time_zones_by_country.xml 中是否存在。

默认时区在 time_zones_by_country.xml 中存在

按照其他的时区 ID 的格式在 timezone.xml 中添加后，再编译版本资源即可验证。

例如需要添加“Africa/Johannesburg”为默认时区，方法如下：

```
<timezones>
...
    <timezone id="Pacific/Fiji">Fiji</timezone>
    <timezone id="Pacific/Tongatapu">Tonga</timezone>
+   <timezone id="Africa/Johannesburg">Johannesburg</timezone>
</timezones>
```


默认时区在 time_zones_by_country.xml 中不存在

需要考虑时区替换。

说明

1. 替换时区时，替换的时区和被替换的时区只能显示一个，请谨慎选择替换的时区。
2. 替换时区时，建议选择系统支持的时区作为替换时区。

时区替换操作步骤如下：

步骤 1 寻找替换时区

例如需要在设置时区列表上添加 Yaounde（雅温德）的时区，但此时区 ID 不在支持的列表中，Yaounde 为喀麦隆首都，则可以考虑使用喀麦隆另一个城市 Douala（杜阿拉）的时区去替换，或者其他国家的与该城市时区规则一样的城市如安哥拉的首都 Luanda（罗安达）的时区去替换。

经过确认 frameworks/base/core/res/res/xml/time_zones_by_country.xml 中包含 Africa/Luanda 时区，而 package/apps/Settings/res/xml-xx/timezones.xml 中没有配置。此时，可以将 Africa/Luanda 时区配置到 timezones.xml 文件中，然后将其显示的内容更换为 Yaounde 的名称，如下所示：

```
<timezones>
...
    <timezone id="Pacific/Fiji">斐济</timezone>
    <timezone id="Pacific/Tongatapu">东加塔布</timezone>
+   <timezone id="Africa/Luanda">雅温德</timezone>
</timezones>
```

步骤 2 添加时区 ID

external/icu/icu4c/source/data/zone 下定义了各个语言下时区名称显示，添加时区 ID 必须按照字母顺序去添加，以修改 en.txt 和 zh.txt 文件为例：

en.txt:

```
zoneStrings{
+   "Africa: Luanda "{
+       ec{"Yaounde"}
+   }
```

zh.txt:

```
"Africa: Luanda "{
-   ec{"罗安达"}
+   ec{"雅温德"}
}
```

其他语言下如有需要也要相应地作修改。

步骤 3 编译 ICU 资源并验证

---结束

4.4.2.2 Android8.1

Android8.0 以及之后的版本，设置中的时区列表是由系统中的 `system/timezone/output_data/android/tzlookup.xml` 文件定义的，该文件就是当前设备支持的所有时区列表，只要在 `tzlookup.xml` 文件有定义的时区 id，都会在 **Setting** 中显示。如果需要新增的时区 `tzlookup.xml` 中不存在，则需要考虑时区替换。

说明

1. 替换时区时，替换的时区和被替换的时区只能显示一个，请谨慎选择替换的时区。
2. 替换时区时，建议选择系统支持的时区作为替换时区。

时区替换操作方法如下：

步骤 1 寻找替换时区

同样以新增 Yaounde 时区为例进行说明。

在 `tzlookup.xml` 中将 Luanda 时区 ID 从安哥拉移动到喀麦隆下面。

```
<!-- 去除安哥拉的时区定义 -->
- <country code="ao" default="Africa/Luanda" everutc="n">
-   <id>Africa/Luanda</id>
- </country>

<!-- 将其移动到喀麦隆的时区定义中 -->
<country code="cm" default="Africa/Douala" everutc="n">
  <id>Africa/Douala</id>
+  <id>Africa/Luanda</id>
</country>
```

步骤 2 添加时区 ID

添加方法请参见“4.4.2.1 步骤 2”。

步骤 3 编译 ICU 资源并验证

---结束

4.4.2.3 Android9.0

Android9.0 平台添加方法同 Android8.0，具体请参见“4.4.2.2 Android8.1”。

4.4.2.4 AndroidQ

时区添加修改时区显示文件，需要进行 ICU 编译后才能验证；而 Google 关闭了 AndroidQ 平台 ICU 编译功能，修改了也无法编译验证，此处不再赘述 AndroidQ 平台相关更改。