在 ZIP64 格式中，**End of Central Directory Record (EOCD)** 的结构与标准 ZIP 不同，用于支持超过 4GB 的大文件。以下是 ZIP64 EOCD 的详细解析和定位方法：

**1. ZIP64 的必要条件**

当以下任一条件满足时，ZIP 文件必须使用 ZIP64 扩展：

* 文件大小 ≥ 4GB
* 文件数量 ≥ 65,535
* Central Directory 偏移量 ≥ 4GB
* 单个文件大小 ≥ 4GB

**2. ZIP64 文件结构关键部分**

完整结构顺序：

复制

下载

[Local File Headers]

[Data Descriptors] (可选)

[Central Directory]

[ZIP64 End of Central Directory Record] ← 新增

[ZIP64 End of Central Directory Locator] ← 新增

[End of Central Directory Record (标准EOCD)]

**3. ZIP64 EOCD 定位步骤**

**(1) 首先找到标准 EOCD**

从文件末尾搜索标准 EOCD 的魔数 0x06054b50（与普通 ZIP 相同）：

cpp

复制

下载

*// 伪代码：从文件末尾向前搜索*

const uint32\_t kEOCDSignature = 0x06054b50;

uint8\_t buffer[4096];

while (file\_offset >= 0) {

read(buffer, sizeof(buffer));

if (find\_signature(buffer, kEOCDSignature)) {

eocd\_offset = calculate\_offset();

break;

}

file\_offset -= buffer\_size - 4; *// 重叠3字节防止跨块*

}

**(2) 检查标准 EOCD 中的 ZIP64 标记**

标准 EOCD 中以下字段若为 0xFFFF，表示需要 ZIP64：

cpp

复制

下载

struct EndOfCentralDirectory {

uint16\_t diskNumber; *// 若为0xFFFF → ZIP64*

uint16\_t startDisk; *// 若为0xFFFF → ZIP64*

uint16\_t numEntriesThisDisk; *// 若为0xFFFF → ZIP64*

uint16\_t totalEntries; *// 若为0xFFFF → ZIP64*

uint32\_t centralDirSize; *// 若为0xFFFFFFFF → ZIP64*

uint32\_t centralDirOffset; *// 若为0xFFFFFFFF → ZIP64*

*// ...其他字段*

};

**(3) 定位 ZIP64 EOCD Locator**

从标准 EOCD 向前 20 字节找到 **ZIP64 EOCD Locator**：

cpp

复制

下载

struct Zip64EndOfCentralDirectoryLocator {

uint32\_t signature; *// 0x07064b50 (固定魔数)*

uint32\_t startDisk; *// ZIP64 EOCD 起始磁盘*

uint64\_t zip64EOCDOffset; *// ← ZIP64 EOCD 的实际偏移量*

uint32\_t totalDisks; *// 磁盘总数*

};

**(4) 读取 ZIP64 EOCD Record**

根据 Locator 提供的偏移量读取 **ZIP64 EOCD Record**：

cpp

复制

下载

struct Zip64EndOfCentralDirectoryRecord {

uint32\_t signature; *// 0x06064b50 (固定魔数)*

uint64\_t recordSize; *// 本记录总大小*

uint16\_t versionMadeBy;

uint16\_t versionNeeded;

uint32\_t diskNumber;

uint32\_t startDisk;

uint64\_t numEntriesThisDisk; *// 实际条目数*

uint64\_t totalEntries; *// 实际总条目数*

uint64\_t centralDirSize; *// Central Directory 实际大小*

uint64\_t centralDirOffset; *// ← Central Directory 实际偏移量*

*// ...扩展字段（可选）*

};

**4. Android 源码中的实现**

Android 在处理大 APK 时通过以下代码适配 ZIP64：

cpp

复制

下载

*// frameworks/base/core/jni/android\_util\_apk/ApkSignatureVerifier.cpp*

status\_t findZip64EndOfCentralDirectory(FILE\* apkFile, off64\_t\* zip64EOCDOffset) {

*// 1. 先找到标准EOCD*

off64\_t eocdOffset;

findEndOfCentralDirectory(apkFile, &eocdOffset);

*// 2. 检查是否需要ZIP64*

EndOfCentralDirectory eocd;

fread(&eocd, 1, sizeof(eocd), apkFile);

if (eocd.centralDirOffset != 0xFFFFFFFF) {

return NOT\_ZIP64; *// 非ZIP64文件*

}

*// 3. 定位ZIP64 Locator（标准EOCD前20字节）*

fseek(apkFile, eocdOffset - 20, SEEK\_SET);

Zip64EndOfCentralDirectoryLocator locator;

fread(&locator, 1, sizeof(locator), apkFile);

*// 4. 验证Locator魔数*

if (locator.signature != 0x07064b50) {

return BAD\_ZIP64;

}

\*zip64EOCDOffset = locator.zip64EOCDOffset;

return OK;

}

**5. 手动解析示例（Python）**

python

复制

下载

import struct

def find\_zip64\_eocd(apk\_path):

with open(apk\_path, 'rb') as f:

*# 1. 找到标准EOCD*

f.seek(-22, 2)

eocd\_data = f.read(22)

if eocd\_data[0:4] != b'PK\x05\x06':

raise ValueError("Invalid EOCD")

*# 2. 检查是否需要ZIP64*

central\_dir\_offset = struct.unpack('<I', eocd\_data[16:20])[0]

if central\_dir\_offset != 0xFFFFFFFF:

return None *# 非ZIP64文件*

*# 3. 读取ZIP64 Locator*

f.seek(-20 - 22, 2) *# 标准EOCD前20字节*

locator\_data = f.read(20)

if locator\_data[0:4] != b'PK\x06\x07':

raise ValueError("Invalid ZIP64 Locator")

zip64\_eocd\_offset = struct.unpack('<Q', locator\_data[8:16])[0]

*# 4. 读取ZIP64 EOCD Record*

f.seek(zip64\_eocd\_offset, 0)

signature = f.read(4)

if signature != b'PK\x06\x06':

raise ValueError("Invalid ZIP64 EOCD")

*# 解析ZIP64 EOCD（略）*

return zip64\_eocd\_offset

**6. 关键数据结构对比**

| **结构** | **魔数** | **关键字段** | **大小** |
| --- | --- | --- | --- |
| **标准 EOCD** | 0x06054b50 | centralDirOffset: uint32 | 22字节+注释 |
| **ZIP64 Locator** | 0x07064b50 | zip64EOCDOffset: uint64 | 20字节 |
| **ZIP64 EOCD** | 0x06064b50 | centralDirOffset: uint64 | 56字节+扩展 |

**7. 实际应用场景**

* **Android 大 APK 处理**：  
  V2/V3 签名块位于 **Central Directory 之前**，需通过 ZIP64 EOCD 获取真实的 centralDirOffset。
* **调试工具**：  
  使用 zipinfo -v 或 xxd 检查 ZIP64 结构：

bash

复制

下载

zipinfo -v large.apk | grep -A 10 "ZIP64"

xxd large.apk | tail -n 50

通过正确解析 ZIP64 结构，Android 系统能够无缝支持超过 4GB 的 APK 文件验证。