收据验证编程指南

Developer

目录

图形、表格和列表

关于收据验证

应用程序收据或 App 内购买项目收据是应用程序的销售记录以及应用程序内任何 App 内购买项目的销售记录。您可以在应用程序中添加收据验证代码，以阻止未经授权的应用程序拷贝运行。请参考许可协议和审核准则以了解相关具体信息，明确您的应用程序为了实现拷贝保护可以以及不可以采取哪些措施。

实现收据验证需要了解密码学以及各种安全编码技术。您必须为自己的应用程序采用独一无二的解决方案，这一点非常重要。

概况一览

验证收据有两种方式：本地验证和通过 App Store 验证。您可以比较这两种方式，并确定哪种方式更适合您的 App 和基础架构。您也可以选择同时实现这两种方式。

本地验证收据

本地验证需要用于读取和验证 PKCS #7 签名的代码，以及用于解析和验证已签署的 Payload 的代码。

相关章节：“本地验证收据 （第 6 页）”、“收据字段 （第 23 页）”

通过 App Store 验证收据

通过 App Store 验证收据需要在您的 App 和服务器之间建立安全连接，并且您的服务器上要有用于通过 App Store 验证收据的代码。

相关章节：“通过 App Store 验证收据 （第 18 页）”、“收据字段 （第 23 页）”

本地验证收据

收据验证应在App启动后，在显示任何用户界面或执行任何子进程前立即执行。调用`NSApplicationMain`函数之前，在`main`函数中执行此检查。为了进一步保证安全，您可以在应用程序运行时定期重复此检查。

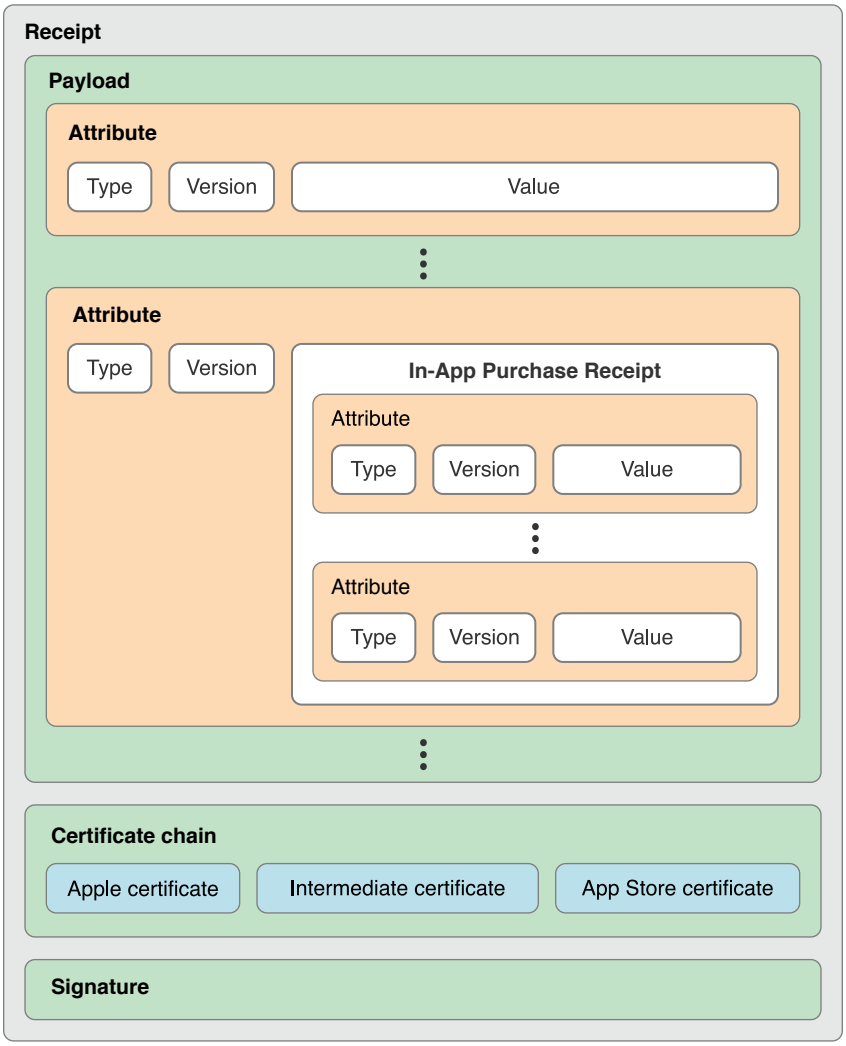
查找和解析收据

从App Store安装应用程序后，该应用程序中会包含一个加密签名的应用程序收据，从而确保只有Apple可以创建有效的收据。该收据存储在应用程序数据包中。调用`NSBundle`类的`appStoreReceiptURL`方法可以找到该收据。

> 注意：在 macOS 中，如果较旧系统上 `appStoreReceiptURL` 方法不可用，您可以转而使用硬编码路径。收据路径为 App 数据包中的 `/Contents/_MASReceipt/receipt`。
>
> 在 iOS 中，如果较旧系统上 `appStoreReceiptURL` 方法不可用，您可以转而通过 App Store 验证 `SKPaymentTransaction` 对象的 `transactionReceipt` 属性。有关详情，请参阅通过 App Store 验证收据 （第 18 页）。

收据是一个有图 1-1中所示结构的二进制文件。

图 **1-1** 收据结构



最外面的部分（在图中标记为*"Receipt"*）是一个由 RFC 2315 定义的 PKCS #7 容器，其中的 Payload 由 ITU-T X.690 定义的 ASN.1（ASN.1 抽象语法标记）编码。Payload 由一系列*"收据属性"*组成。每个收据属性包含一个类型、一个版本和一个值。

Payload 的结构由“列表 1-1”中的 ASN.1 标记定义。您可以结合使用此定义和 `asn1c` 工具，生成用于解码 Payload 的数据类型声明和函数，而不必手动编写这一部分代码。您可能需要先安装 `asn1c`（可通过 MacPorts 和 SourceForge 获得）。

有关收据中 Key 的更多信息，请参阅“收据字段 （第 23 页）”。

若要生成代码，需要将“列表 1-1”中所示的 Payload 说明保存到文件中，然后在“终端”内运行以下命令：

```
asn1c -fnative-types filename
```

`asn1c` 工具完成在当前目录下的文件生成后，将它生成的文件添加到您的 Xcode 项目中。

列表 **1-1** Payload 格式的 ASN.1 定义

```
ReceiptModule DEFINITIONS ::=
BEGIN

ReceiptAttribute ::= SEQUENCE {
    type    INTEGER,
    version INTEGER,
    value   OCTET STRING
}

Payload ::= SET OF ReceiptAttribute

END
```

计算 GUID 的哈希（Hash）

在 macOS 中，使用在 macOS 中获取 GUID （第 12 页） 所述的方法获取电脑的 GUID。

在 iOS 中，使用 `UIDevice` 的 `identifierForVendor` 属性的返回值作为电脑的 GUID。

要计算哈希（Hash），首先将 GUID 值与不透明值（类型 4 的属性）和 Bundle Identifier（数据包标识符）连接在一起。使用收据中的原始字节，不要进行任何 UTF-8 字符串解释或标准化。然后，计算连接的这一系列字节的 SHA-1 哈希（Hash）。

验证收据

要验证收据，请按顺序执行以下测试：

1. 找到收据。

   如果没有收据，则验证失败。

2. 验证收据是否由 Apple 正确签署

   如果收据不是由 Apple 签署，则验证失败。

3. 验证收据中的 Bundle Identifier（数据包标识符）与在 `Info.plist` 文件中含有您要的 `CFBundleIdentifier` 值的硬编码常量相匹配。

   如果两者不匹配，则验证失败。

4. 验证收据中的版本标识符字符串与在 `Info.plist` 文件中含有您要的 `CFBundleShortVersionString` 值（macOS）或 CFBundleVersion 值（iOS）的硬编码常量相匹配。

   如果两者不匹配，则验证失败。

5. 按照“计算 GUID 的哈希（Hash）（第 8 页）”所述计算 GUID 的哈希（Hash）。

   如果结果与收据中的哈希（Hash）不匹配，则验证失败。

如果通过所有测试，则验证成功。

> 注意：Bundle Identifier（数据包标识符）和版本标识符字符串是 UTF-8 字符串，而不仅仅是一系列字节。确保您相应地编写比较逻辑代码。

如果您的 App 支持“批量购买计划”，请检查收据的有效日期。

响应收据验证失败

验证可能由于各种原因而失败。例如，当用户将应用程序从一台 Mac 拷贝到另一台 Mac 时，由于 GUID 不再匹配，导致收据验证失败。

在 macOS 中验证失败时退出

如果在 macOS 中验证失败，请调用状态为 `173` 的 `exit`。此退出状态会通知系统，您的应用程序已确定收据无效。此时，系统会尝试获取有效收据，并可能提示用户提供 iTunes 凭据。

如果系统成功获取有效收据，则会重新启动应用程序。否则，会向用户显示错误消息，解释出现问题的原因。

验证失败时，不要向用户显示任何错误消息。系统将负责尝试获取有效收据或通知用户收据无效。

在 iOS 中验证失败时刷新收据

如果在 iOS 中验证失败，请使用 `SKReceiptRefreshRequest` 类刷新收据。

不要尝试终止 App。您可以选择：为用户提供宽限期，或限制 App 内的功能。

为 Mac App 设置最低系统版本

在您应用程序的 `Info.plist` 文件中为 `LSMinimumSystemVersion` Key 包含 10.6.6 或更大的值。如果收据在 10.6.6 版本或之前版本的 macOS 上验证失败，您的应用程序会在启动后立即退出，并且不向用户做出任何解释。早期版本的 macOS 不解释退出状态 `173`，因此不会尝试获取有效收据或显示任何错误消息。

请勿本地化版本号

如果您的应用程序已本地化，则 `CFBundleShortVersionString` Key 不应在您的应用程序的任何 `InfoPlist.strings` 文件中出现。收据中存储的是您的 `Info.plist` 文件中未本地化的值，尝试对此 Key 的值进行本地化会导致收据验证失败。

保护验证检查的安全

攻击者可能会尝试通过修补应用程序二进制文件或更改验证代码所依赖的基本操作系统例程来绕开验证码。抵御这类的攻击需要多种编码技术，其中包括：

- 嵌入加密检查所用的代码，而不是使用系统提供的 API
- 避免使用简单的代码结构，因为这会为修补应用程序二进制文件提供细节目标

  例如，避免编写如下代码：

```
if (failedValidation) {
   exit(173);
}
```

- 实施代码鲁棒性技术，如代码混淆

如果多个应用程序使用相同的代码执行验证，则此通用代码签名会成为应用程序二进制文件修补工具的目标。

- 请确保，即使 `exit` 函数未能终止您的应用程序，您的应用程序也会停止运行

在开发过程中进行测试

为了在开发过程中测试您的主要应用程序，您必须具备有效收据，以用于启动应用程序。若要进行此项设置，请执行以下操作：

1. 确保您有互联网访问权限，以连接 Apple 的服务器。
2. 连按应用程序以启动应用程序（或以其他途径，通过“启动服务”来启动应用程序）。

启动应用程序后，会发生以下情况：

- 由于不存在收据，您的应用程序验证收据失败并退出（状态为 `173`）
- 系统解读退出状态并尝试获取有效收据。系统假设您的应用程序签名证书有效，并为应用程序安装有效的收据。系统可能会提示您提供 iTunes 凭证
- 系统重新启动您的应用程序，并且您的应用程序成功验证收据

当此开发收据安装后，您可以通过任何方法启动应用程序，例如使用 `gdb` 或 Xcode 调试工具。

验证 App 内购买项目

若要验证 App 内购买项目，您的应用程序要按顺序执行以下测试：

1. 如前几节所述，解析并验证应用程序的收据。

   如果收据无效，则所有 App 内购买项目均为无效。

2. 解析 App 内购买项目收据（类型为 17 的属性的值）。

   与应用程序的收据一样，每个 App 内购买项目收据包含一组属性。这些收据的结构在“列表 1-2 （第 12 页）”中定义。在解析收据时，您可以使用 `asn1c` 工具，通过 ASN.1 描述生成一些代码。忽略所有表中没有出现的类型的属性，这些类型的属性保留供系统使用，并且它们的内容随时可能更改。

   有关收据中字段的更多信息，请参阅“收据字段 （第 23 页）”。

3. 检查每个 App 内购买项目收据的产品标识符，并启用您 App 中的相应功能或内容。若要了解更多关于如何计算订阅有效时间段的信息，请参阅“处理订阅”。

如果 App 内购买项目收据的验证失败，您的应用程序将不会启用功能或内容。

列表 1-2　App 内购买项目收据格式的 ASN.1 定义

```
InAppAttribute ::= SEQUENCE {
    type                    INTEGER,
    version                 INTEGER,
    value                   OCTET STRING
}

InAppReceipt ::= SET OF InAppAttribute
```

当重新下载购买项目时，系统将使用原始交易标识符属性和原始交易日期属性。重新下载的购买项目将被给予一个新的交易标识符，但其中包含原始购买项目的标识符和日期。

收据验证实现提示

本节提供一些代码列表，供您在实现收据验证时参考。

在 macOS 中获取 GUID

在 macOS 中，按照列表 1-3 中展示的样式（或使用完全相同的代码）获取 GUID。这样可以确保您在验证代码中获取 GUID 所用的方法与创建该应用程序的收据时所用的方法完全相同。

列表 1-3　获取电脑的 GUID

```
#import <IOKit/IOKitLib.h>
#import <Foundation/Foundation.h>

// Returns a CFData object, containing the computer's GUID.
CFDataRef copy_mac_address(void)
{
    kern_return_t             kernResult;
    mach_port_t               master_port;
    CFMutableDictionaryRef    matchingDict;
    io_iterator_t             iterator;
    io_object_t               service;
    CFDataRef                 macAddress = nil;
```

```
    kernResult = IOMasterPort(MACH_PORT_NULL, &master_port);
    if (kernResult != KERN_SUCCESS) {
        printf("IOMasterPort returned %d\n", kernResult);
        return nil;
    }

    matchingDict = IOBSDNameMatching(master_port, 0, "en0");
    if (!matchingDict) {
        printf("IOBSDNameMatching returned empty dictionary\n");
        return nil;
    }

   kernResult = IOServiceGetMatchingServices(master_port, matchingDict, &iterator);
    if (kernResult != KERN_SUCCESS) {
        printf("IOServiceGetMatchingServices returned %d\n", kernResult);
        return nil;
    }

    while((service = IOIteratorNext(iterator)) != 0) {
        io_object_t parentService;

        kernResult = IORegistryEntryGetParentEntry(service, kIOServicePlane,
                &parentService);
        if (kernResult == KERN_SUCCESS) {
            if (macAddress) CFRelease(macAddress);

           macAddress = (CFDataRef) IORegistryEntryCreateCFProperty(parentService,
                    CFSTR("IOMACAddress"), kCFAllocatorDefault, 0);
            IOObjectRelease(parentService);
        } else {
            printf("IORegistryEntryGetParentEntry returned %d\n", kernResult);
        }

        IOObjectRelease(service);
```

```
    }
    IOObjectRelease(iterator);

    return macAddress;
}
```

解析收据并验证签名

在您使用 OpenSSL 和 `asn1c` 实现收据验证时，可以使用以下代码列表作为大纲来参照。为了指导您编写自己的代码，以下代码列表突出了相关的 API 和数据结构，请不要直接拷贝粘贴。

如果您使用 OpenSSL，请将二进制文件与 OpenSSL 静态链接。动态链接 OpenSSL 已被弃用，并会导致构建版本警告。

确保您的代码按照列表中所述完成以下任务：

1. 验证签名（“列表 1-4 （第 14 页）”）。
2. 解析 Payload（“列表 1-5 （第 15 页）”）。
3. 提取收据属性（“列表 1-6 （第 16 页）”）。
4. 计算 GUID 的哈希（Hash）（“列表 1-7 （第 17 页）”）。

列表 **1-4**　使用 OpenSSL 验证签名

```
/* The PKCS #7 container (the receipt) and the output of the verification. */
BIO *b_p7;
PKCS7 *p7;

/* The Apple root certificate, as raw data and in its OpenSSL representation. */
BIO *b_x509;
X509 *Apple;

/* The root certificate for chain-of-trust verification. */
X509_STORE *store = X509_STORE_new();

/* ... Initialize both BIO variables using BIO_new_mem_buf() with a buffer and its
 size ... */
```

```
/* Initialize b_out as an output BIO to hold the receipt payload extracted during
 signature verification. */
BIO *b_out = BIO_new(BIO_s_mem());


/* Capture the content of the receipt file and populate the p7 variable with the
PKCS #7 container. */
p7 = d2i_PKCS7_bio(b_p7, NULL);


/* ... Load the Apple root certificate into b_X509 ... */


/* Initialize b_x509 as an input BIO with a value of the Apple root certificate and
 load it into X509 data structure. Then add the Apple root certificate to the
structure. */
Apple = d2i_X509_bio(b_x509, NULL);
X509_STORE_add_cert(store, Apple);


/* Verify the signature. If the verification is correct, b_out will contain the
PKCS #7 payload and rc will be 1. */
int rc = PKCS7_verify(p7, NULL, store, NULL, b_out, 0);


/* You must verify the fingerprint of the root certificate and verify the OIDs of
 the intermediate certificate and signing certificate. The OID in the certificate
 policies extension of the intermediate certificate is (1 2 840 113635 100 6 2 1),
 and the marker OID of the signing certificate is (1 2 840 113635 100 6 11 1). */
```

列表 **1-5**　使用 asn1c 解析 Payload

```
#include "Payload.h" /* This header file is generated by asn1c. */


/* The receipt payload and its size. */
void *pld = NULL;
size_t pld_sz;


/* Variables used to parse the payload. Both data types are declared in Payload.h.
 */
Payload_t *payload = NULL;
asn_dec_rval_t rval;
```

```
/* ... Load the payload from the receipt file into pld and set pld_sz to the payload
 size ... */


/* Parse the buffer using the decoder function generated by asn1c.  The payload
variable will contain the receipt attributes. */
rval = asn_DEF_Payload.ber_decoder(NULL, &asn_DEF_Payload, (void **)&payload, pld,
 pld_sz, 0);
```

列表 **1-6**　　提取收据属性

```
/* Variables used to store the receipt attributes. */
OCTET_STRING_t *bundle_id = NULL;
OCTET_STRING_t *bundle_version = NULL;
OCTET_STRING_t *opaque = NULL;
OCTET_STRING_t *hash = NULL;


/* Iterate over the receipt attributes, saving the values needed to compute the
GUID hash. */
size_t i;
for (i = 0; i < payload->list.count; i++) {
    ReceiptAttribute_t *entry;


    entry = payload->list.array[i];


    switch (entry->type) {
        case 2:
            bundle_id = &entry->value;
            break;
        case 3:
            bundle_version = &entry->value;
            break;
        case 4:
            opaque = &entry->value;
            break;
        case 5:
            hash = &entry->value;
```

```
            break;
        }
    }
```

列表 **1-7**　　计算 GUID 的哈希（Hash）

```
/* The GUID returned by copy_mac_address() is a CFDataRef.  Use CFDataGetBytePtr()
 and CFDataGetLength() to get a pointer to the bytes that make up the GUID and to
 get its length. */
UInt8 *guid = NULL;
size_t guid_sz;

/* Declare and initialize an EVP context for OpenSSL. */
EVP_MD_CTX evp_ctx;
EVP_MD_CTX_init(&evp_ctx);

/* A buffer for result of the hash computation. */
UInt8 digest[20];

/* Set up the EVP context to compute a SHA-1 digest. */
EVP_DigestInit_ex(&evp_ctx, EVP_sha1(), NULL);

/* Concatenate the pieces to be hashed.  They must be concatenated in this order.
 */
EVP_DigestUpdate(&evp_ctx, guid, guid_sz);
EVP_DigestUpdate(&evp_ctx, opaque->buf, opaque->size);
EVP_DigestUpdate(&evp_ctx, bundle_id->buf, bundle_id->size);

/* Compute the hash, saving the result into the digest variable. */
EVP_DigestFinal_ex(&evp_ctx, digest, NULL);
```

通过 App Store 验证收据

使用受信任的服务器与 App Store 通信。通过使用自己的服务器，您可以将 App 设计为仅识别和信任您的服务器，并且您可以确保您的服务器与 App Store 服务器的连接。您不可能直接在用户的设备和 App Store 之间建立可靠连接，因为该连接的两端都不受您控制。

与 App Store 的通信采用 JSON 字典结构（如 RFC 4627 中所定义）。二进制文件数据采用 base64 编码（如 RFC 4648 中所定义）。

读取收据数据

若要获取收据数据，请使用 `NSBundle` 的 `appStoreReceiptURL` 方法查找 App 的收据，然后读取整个文件。如果无法使用 `appStoreReceiptURL` 方法，您也可以转而使用交易的 `transactionReceipt` 属性值（用于实现向后兼容）。然后将此数据发送到您的服务器（与跟服务器的所有其他互动一样，您需要对细节负责）。

```
// Load the receipt from the app bundle.
NSURL *receiptURL = [[NSBundle mainBundle] appStoreReceiptURL];
NSData *receipt = [NSData dataWithContentsOfURL:receiptURL];
if (!receipt) { /* No local receipt -- handle the error. */ }

/* ... Send the receipt data to your server ... */
```

向 App Store 发送收据数据

在您的服务器上，使用以下 Key 创建一个 JSON 对象：

Key	值
`receipt-data`	base64 编码的收据数据。
`password`	*仅用于包含自动续期订阅的收据。*您 App 的共享密钥（十六进制字符串）。

Key	值
exclude-old-transactions	*仅用于包含自动续期订阅或非续期订阅的iOS 7样式的App收据。* 如果值为 true，仅响应包括所有订阅的最新续期交易。

将此 JSON 对象作为 HTTP POST 请求的 Payload 提交。在测试环境中，使用 `https://sandbox.itunes.apple.com/verifyReceipt` 作为网址（URL）。在生产环境中，使用 `https://buy.itunes.apple.com/verifyReceipt` 作为网址（URL）。

```
NSData *receipt; // Sent to the server by the device

// Create the JSON object that describes the request
NSError *error;
NSDictionary *requestContents = @{
    @"receipt-data": [receipt base64EncodedStringWithOptions:0]
};
NSData *requestData = [NSJSONSerialization dataWithJSONObject:requestContents
                                                      options:0
                                                        error:&error];

if (!requestData) { /* ... Handle error ... */ }

// Create a POST request with the receipt data.
NSURL *storeURL = [NSURL
URLWithString:@"https://buy.itunes.apple.com/verifyReceipt"];
NSMutableURLRequest *storeRequest = [NSMutableURLRequest requestWithURL:storeURL];
[storeRequest setHTTPMethod:@"POST"];
[storeRequest setHTTPBody:requestData];

// Make a connection to the iTunes Store on a background queue.
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
[NSURLConnection sendAsynchronousRequest:storeRequest queue:queue
        completionHandler:^(NSURLResponse *response, NSData *data, NSError
*connectionError) {
    if (connectionError) {
        /* ... Handle error ... */
```

```
    } else {
        NSError *error;
        NSDictionary *jsonResponse = [NSJSONSerialization JSONObjectWithData:data
 options:0 error:&error];
        if (!jsonResponse) { /* ... Handle error ...*/ }
        /* ... Send a response back to the device ... */
    }
}];
```

解析响应

响应的 Payload 是一个 JSON 对象，包含以下 Key 和值：

Key	值
status	值为 0（如果收据有效）或表 2-1 （第 21 页）中所列的错误代码之一。 对于 iOS 6 样式的交易收据，状态代码反映特定交易收据的状态。 对于 iOS 7 样式的 App 收据，状态代码反映 App 收据的整体状态。例如，如果您发送一个有效 App 收据并且其中包含一个已过期订阅，则响应为 0，因为收据的整体状态为有效。
receipt	已发送的供验证收据的 JSON 表现形式。有关收据中 Key 的更多信息，请参阅“收据字段 （第 23 页）”。
latest_receipt	*只有在交易收据为 iOS 6 样式且为自动续期订阅时才会返回。*最新续期的 base-64 编码收据。
latest_receipt_info	*只有在交易收据为 iOS 6 样式且为自动续期订阅时才会返回。*最新续期的收据的 JSON 表现形式。
latest_expired_- receipt_info	*只有在交易收据为 iOS 6 样式且为自动续期订阅时才会返回。*已过期订阅的收据的 JSON 表现形式。

Key	值
pending_renewal_info	*只有在 App 收据为 iOS 7 样式且包含自动续期订阅时才会返回。*在 JSON 文件中，该 Key 的值是一个数组，数组中的每个元素都包含每个自动续期订阅的待处理续期信息，这些自动续期订阅由产品标识符 （第 26 页）标识。待处理续期可能是指安排在将来进行的续期或过去由于某些原因失败的续期。
is-retryable	重新验证该收据。*仅适用于状态码 21100-21199（列于*表 2-1 （第 21 页））*）*

表 2-1 状态代码

状态代码	描述
21000	App Store 无法读取您提供的 JSON 对象。
21002	receipt-data 属性中的数据格式错误或丢失。
21003	无法认证收据。
21004	您提供的共享密钥与您帐户存档的共享密钥不匹配。
21005	收据服务器当前不可用。
21006	此收据有效，但订阅已过期。当此状态代码返回到您的服务器时，收据数据也将解码并作为响应的一部分返回。 *只有在交易收据为 iOS 6 样式且为自动续期订阅时才会返回。*
21007	此收据来自测试环境，但发送到生产环境进行验证。应将其发送到测试环境。
21008	此收据来自生产环境，但发送到测试环境进行验证。应将其发送到生产环境。
21010	此收据无法获得授权。对待此收据的方式与从未进行过任何交易时的处理方式相同。
21100-21199	内部数据访问错误。

在检查自动续期订阅当前是否有效时，latest_receipt 和 latest_receipt_info Key 的值非常有用。

在检查自动续期订阅是否过期时，latest_expired_receipt_info Key 的值非常有用。使用该值并结合“订阅到期意图 （第 28 页）”中的值获取订阅过期的原因。

pending_renewal_info Key 的值对于获取自动续期订阅的待续期交易的关键信息非常有用。

通过提供 App 收据或为订阅提供交易收据，并检查收据中的值，您可以获取当前有效订阅时间段的信息。如果正在验证的是针对最新续期的收据，则 `latest_receipt` 的值与请求中 `receipt-data` 的值相同，且 `latest_receipt_info` 的值与 `receipt` 的值相同。

收据字段

收据由一些字段组成。有些字段仅用于本地验证，在 ASN.1 形式的收据中出现；有些字段仅用于通过 App Store 验证，在 JSON 形式的收据中出现。下文未记录的 Key 是保留供 Apple 使用的，您的 App 必须忽略这些 Key。

App 收据字段

Bundle Identifier（数据包标识符）

App 的 Bundle Identifier（数据包标识符）。

ASN.1 Field Type 2

ASN.1 Field Value UTF8STRING

JSON Field Name bundle_id

JSON Field Value 字符串

该值与 `Info.plist` 文件中的 `CFBundleIdentifier` 的值相对应。使用此值验证收据是否的确由您的 App 生成。

App 版本

App 的版本号。

ASN.1 Field Type 3

ASN.1 Field Value UTF8STRING

JSON Field Name application_version

JSON Field Value 字符串

该值与 `Info.plist` 中的 `CFBundleVersion`（iOS 中）或 `CFBundleShortVersionString`（macOS 中）的值相对应。

不透明值

不透明值用于在验证期间结合其他数据计算 SHA-1 哈希（Hash）。

ASN.1 Field Type 4

ASN.1 Field Value 一系列字节

JSON Field Name（无）

JSON Field Value（无）

SHA-1 哈希（Hash）

用于验证收据的 SHA-1 哈希（Hash）。

ASN.1 Field Type 5

ASN.1 Field Value 20 字节 SHA-1 摘要

JSON Field Name（无）

JSON Field Value（无）

App 内购买项目收据

App 内购买项目的收据。

ASN.1 Field Type 17

ASN.1 Field Value 一组 App 内购买项目收据属性

JSON Field Name in_app

JSON Field Value App 内购买项目收据数组

在 JSON 文件中，此 Key 的值是一个包含所有 App 内购买项目收据的数组。在 ASN.1 文件中，存在多个类型为 17 的字段，其中每个字段都包含一个单独的 App 内购买项目收据。

> 注意：空数组为有效收据。

在进行消耗型产品购买时，该 App 内购买项目收据被添加到收据中。在您的 App 完成该笔交易之前，该 App 内购买项目收据会一直保留在收据内。在交易完成后，该 App 内购买项目收据会在下次收据更新时（例如，顾客进行下一次购买，或者您的 App 明确刷新收据时）从收据中移除。

非消耗型产品、自动续期订阅、非续期订阅或免费订阅的 App 内购买项目收据无限期保留在收据中。

原始应用程序版本

最初购买的 App 的版本。

ASN.1 Field Type 19

ASN.1 Field Value UTF8STRING

JSON Field Name original_application_version

JSON Field Value 字符串

此值与最初进行购买时`Info.plist` 文件中的 `CFBundleVersion`（iOS 中）或 `CFBundleShortVersionString`（macOS 中）的值相对应。

在沙箱技术环境中，这个字段值始终为“1.0”。

收据创建日期

App 收据的创建日期。

ASN.1 Field Type 12

ASN.1 Field Value IA5STRING，被视作一个 RFC 3339 日期

JSON Field Name creation_date

JSON Field Value IA5STRING，被视作一个 RFC 3339 日期

验证收据时，使用这个日期验证收据的签名。

> 注意：许多加密算法库在验证 PKCS7 数据包时，默认使用设备当前的时间和日期，但在验证收据签名时，这种做法可能无法得到正确的结果。例如，如果收据是使用有效证书签署的，但证书已过期，则错误使用设备的当前日期进行验证会返回无效结果。
>
> 因此，应当确保您的 App 始终使用“收据创建日期”字段中的日期对收据签名进行验证。

收据有效日期

App 收据的到期日期。

ASN.1 Field Type 21

ASN.1 Field Value IA5STRING，被视作一个 RFC 3339 日期

JSON Field Name expiration_date

JSON Field Value IA5STRING，被视作一个 RFC 3339 日期

此 Key 仅适用于通过“批量购买计划”购买的 App。如果没有此 Key，则收据就不会过期。

验证收据时，通过比较该日期与当前日期来确定收据是否过期。不要试图使用此日期来计算任何其他信息，例如到期前的剩余时间。

App 内购买项目收据字段

数量

购买的项目的数量。

ASN.1 Field Type 1701

ASN.1 Field Value INTEGER

JSON Field Name quantity

JSON Field Value 字符串，被视作一个整数

这个值与存储在交易的 `payment` 属性中 `SKPayment` 对象的 `quantity` 属性相对应。

产品标识符

所购项目的产品标识符。

ASN.1 Field Type 1702

ASN.1 Field Value UTF8STRING

JSON Field Name product_id

JSON Field Value 字符串

这个值与存储在交易的 `payment` 属性中 `SKPayment` 对象的 `productIdentifier` 属性相对应。

交易标识符

所购项目的交易标识符。

ASN.1 Field Type 1703

ASN.1 Field Value UTF8STRING

JSON Field Name transaction_id

JSON Field Value 字符串

这个值与交易的 `transactionIdentifier` 属性相对应。

针对恢复先前交易的交易，此值与原始购买交易的交易标识符不同。在自动续期订阅收据中，每次在新设备上自动续期或恢复订阅时，都会为交易标识符生成新值。

原始交易标识符

针对恢复先前交易的交易，该交易标识符代表原始交易。否则，与交易标识符相同。

ASN.1 Field Type 1705

ASN.1 Field Value UTF8STRING

JSON Field Name original_transaction_id

JSON Field Value 字符串

这个值与原始交易的 `transactionIdentifier` 属性相对应。

为特定订阅生成的所有收据中此值相同。此值可用于将同一位个人顾客订阅的多个 iOS 6 样式交易收据相关联。

购买日期

购买项目的日期和时间。

ASN.1 Field Type 1704

ASN.1 Field Value IA5STRING，被视作一个 RFC 3339 日期

JSON Field Name purchase_date

JSON Field Value 字符串，被视作一个 RFC 3339 日期

这个值与交易的 `transactionDate` 属性相对应。

对于恢复先前交易的交易，购买日期与原始购买日期相同。通过"原始购买日期 （第 28 页）"字段可获取原始交易的日期。

在自动续期订阅收据中，购买日期是购买或续期订阅的日期（无论订阅是否失效）。对于在当前时间段有效日期发生的自动续期，购买日期与当前时段的结束日期相同，为下一时间段的开始日期。

原始购买日期

对于恢复先前交易的交易，此值为原始交易的日期。

ASN.1 Field Type 1706

ASN.1 Field Value IA5STRING，被视作一个 RFC 3339 日期

JSON Field Name original_purchase_date

JSON Field Value 字符串，被视作一个 RFC 3339 日期

这个值与原始交易的 `transactionDate` 属性相对应。

在自动续期订阅收据中，此值表示订阅时间段的开始（即使订阅已续期）。

订阅有效日期

订阅的有效日期，以距离格林威治时间 1970 年 1 月 1 日 00 时 00 分 00 秒的毫秒数表示。

ASN.1 Field Type 1708

ASN.1 Field Value IA5STRING，被视作一个 RFC 3339 日期

JSON Field Name expires_date

JSON Field Value 字符串，被视作一个 RFC 3339 日期

此Key仅适用于自动续期订阅收据。使用此值确定订阅续期日期或有效日期，确认顾客是否拥有对内容或服务的访问权限。验证最新收据后，如果最新续期交易的订阅有效日期是过去的日期，则可以断定订阅已过期。

订阅到期意图

针对已过期订阅，解释订阅过期的原因。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name expiration_intent

JSON Field Value 字符串，被视作一个整数

”1“- 顾客取消订阅。

”2“- 账单错误；例如，顾客的付款信息不再有效。

“3”- 顾客不同意近期提价。

“4”- 在续期时，产品暂无供应。

“5”- 未知错误。

此 Key 仅适用于包含已过期自动续期订阅的收据。您可以使用此值决定是否在您的 App 中显示适当的消息，以便顾客重新订阅。

订阅重试旗标

针对已过期订阅，显示 Apple 是否仍尝试自动续期订阅。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name is_in_billing_retry_period

JSON Field Value 字符串，被视作一个整数

“1”- App Store 仍然尝试续期订阅。

“0”- App Store 已停止尝试续期订阅。

此 Key 仅适用于自动续期订阅收据。如果由于 App Store 无法完成交易导致顾客的订阅未能续期，此值将反映 App Store 是否仍然尝试续期订阅。

订阅试用期

反映订阅是否处于“免费试用”期。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name is_trial_period

JSON Field Value 字符串

此 Key 仅适用于自动续期订阅收据。当顾客的订阅目前处于免费试用期时，此 Key 的值为“`true`”，否则为"`false`"。

取消日期

针对由 Apple 顾客支持取消的交易，反映取消交易的时间和日期。

ASN.1 Field Type 1712

ASN.1 Field Value IA5STRING，被视作一个 RFC 3339 日期

JSON Field Name cancellation_date

JSON Field Value 字符串，被视作一个 RFC 3339 日期

对已取消的收据的处理方式与未进行任何交易时相同。

> 注意：取消的 App 内购买项目将无限期保留在收据中。仅在为非消耗型产品、自动续期订阅、非续期订阅或免费订阅退款时适用。

取消原因

针对已取消的交易，反映取消的原因。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON FieldName cancellation_reason

JSON Field Value 字符串，被视作一个整数

“1”- 顾客取消交易，因为您的 App 确实存在问题或顾客认为存在问题。

“0”- 由于其他原因取消交易，例如顾客意外地进行了此次交易。

结合取消日期使用此值，确定您 App 内存在的可能导致顾客联系 Apple 顾客支持的问题。

App 项目 ID

一个字符串，App Store 用它来唯一识别创建交易的应用程序。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name app_item_id

JSON Field Value 字符串

如果您的服务器支持多个应用程序，您可以使用此值来区分它们。

只有在生产环境中，系统才会为 App 分配标识符，因此该 Key 不适用于在测试环境中创建的收据。

此字段不适用于 Mac App。

另请参阅"Bundle Identifier（数据包标识符）（第 23 页）"。

外部版本标识符

一个任意数字，用于唯一标识应用程序版本。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name version_external_identifier

JSON Field Value 字符串

此 Key 不适用于在测试环境中创建的收据。使用此 Key 识别顾客购买的 App 版本。

网络订单行项目 ID

用于识别订阅购买项目的首要 Key。

ASN.1 Field Type 1711

ASN.1 Field Value INTEGER

JSON Field Name web_order_line_item_id

JSON Field Value 字符串

此值是用于识别设备间购买事件（包括订阅续期购买事件）的唯一 ID。

订阅自动续期状态

自动续期订阅的当前续期状态。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name auto_renew_status

JSON Field Value 字符串，被视作一个整数

“1”- 订阅将在当前订阅时间段结束时续期。

“0”- 顾客已关闭自动续期订阅。

此 Key 仅适用于自动续期订阅收据，包括有效订阅或已过期订阅。此 Key 的值不应被视作顾客的订阅状态。您可以使用此值在 App 中显示备用订阅产品（例如较低等级订阅计划，以供顾客降级当前的订阅计划）。

订阅自动续期偏好

自动续期订阅的当前续期偏好。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name auto_renew_product_id

JSON Field Value 字符串

此 Key 仅适用于自动续期订阅收据。此 Key 的值与顾客订阅续期的产品的 `productIdentifier` 属性相对应。您可以在当前订阅到期前，使用此值向顾客提供替代的服务级别。

订阅价格认同状态

顾客当前对订阅价格提价的价格认同状态。

ASN.1 Field Type（无）

ASN.1 Field Value（无）

JSON Field Name price_consent_status

JSON Field Value 字符串，被视作一个整数

“1”- 顾客同意提价。订阅将以提价后的价格续期。

“0”- 顾客没有针对提价一事采取任何措施。如果顾客在续期日期之前不采取任何措施，订阅将到期。

此 Key 只在当订阅价格已提价，并且不为现有订阅者保留现有价格时，适用于自动续期订阅收据。您可以使用此值跟踪顾客对新价格的接受情况，并相应地采取措施。

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Draft copy. Trademarks go here.

同时在美国和加拿大出版。