# 青莲云热补丁升级操作流程

| 版本 | 编写/修订说明 | 修订人 | 修订日期 | 备注 |
| --- | --- | --- | --- | --- |
| 1.0.0 | 创建文档 | 杨思嘉 | 20190605 | |

# 目录

# 1 前言

热补丁（hotfix），又称为 patch，指能够修复软件漏洞的一些代码，是一种快速、低成本修复产品软件版本缺陷的方式。热补丁功能支持用户在不重启当前系统的情况下，完成云端批量推送固件补丁/脚本/执行文件到设备打补丁的功能。常用场景为带 OS 系统的设备，升级一下组件、库、资源等等。

# 2 概述

设备上线后，每六个小时会向云端请求获取已发布的未下载补丁列表，用户可自行选择哪些补丁需要被下载。补丁下载完成后将执行结果告知云端，完成热补丁升级流程。

即热补丁升级流程包含以下五个步骤

1) 云端发布新版本补丁

2) 设备端拉取补丁列表

3) 设备端请求下载

4) 设备端下载补丁数据

5) 设备端上传补丁执行结果

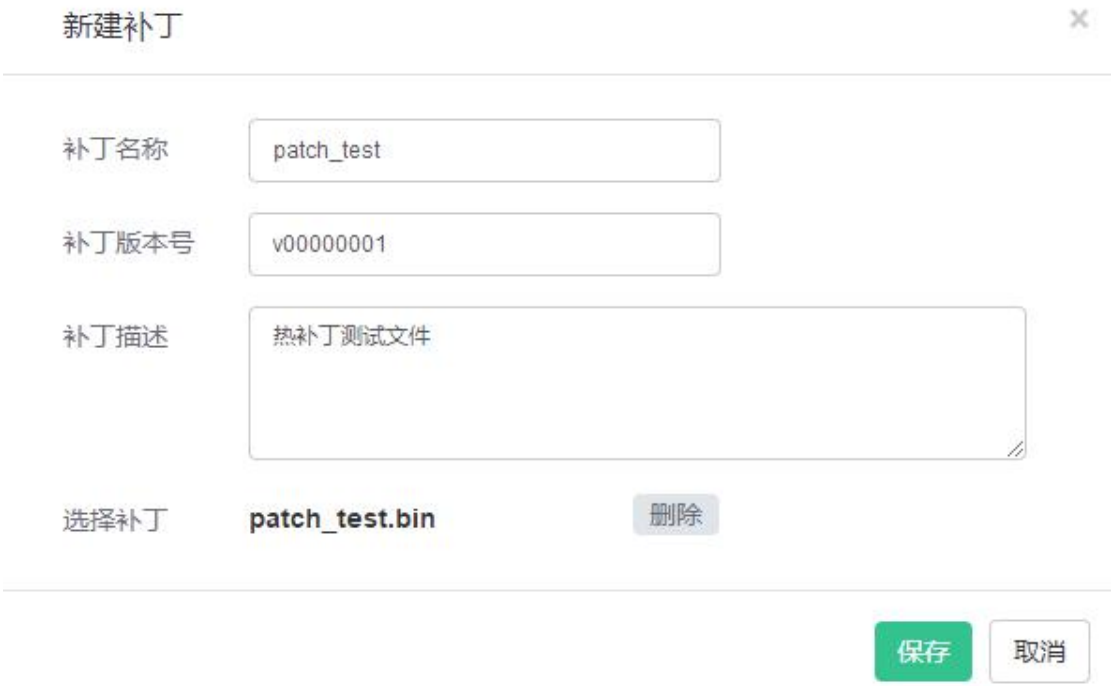
下面详细介绍各步骤的操作流程。

# 3 补丁发布

## 3.1 上传补丁

进入控制台的补丁升级页面，点击右上角的 

按钮，填写如下"新建补丁"对话框。



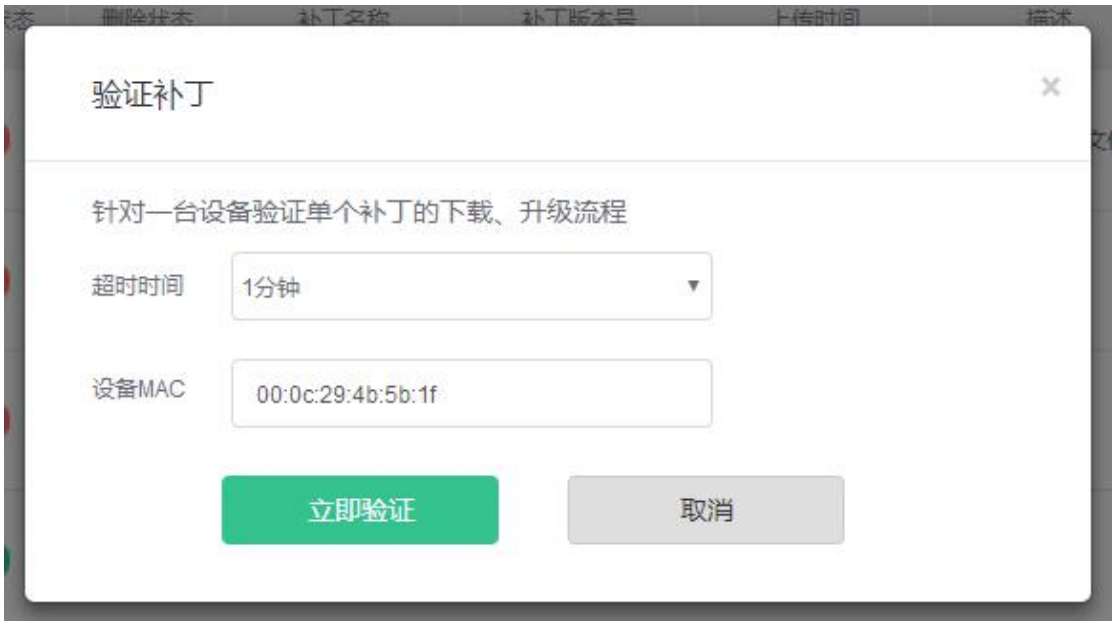

- 补丁名称：可根据补丁特性填入合适的名称，长度不超过 16 字节。
- 补丁版本号：用户自定义格式，长度不超过 16 字节，此字段同一产品下唯一。
- 补丁描述，可根据补丁特性填入合适的描述信息
- 选择补丁：上传用于升级的补丁文件，支持后缀名为"bin"或"zip"格式的文件。

## 3.2 验证补丁

为保证在线设备下载新补丁后，补丁能够正常工作，要先对该补丁进行验证。验证成功才可进入试发布阶段。

设备要支持热补丁功能，先要对设备端进行编码，编码涉及的 patch 接口请参考《青莲云嵌入式 SDK 开发使用文档》第 5.2 章节。

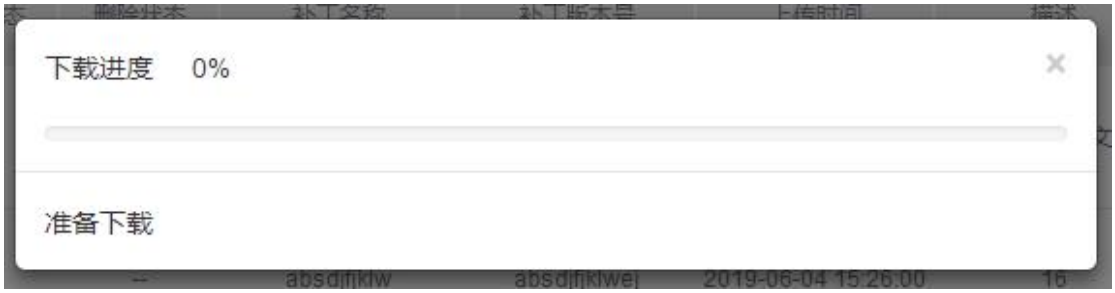
点击补丁版本右边的 验证补丁 按钮，弹出“验证补丁”对话框，如下图所示：




选择一个补丁验证超时的时长，云端会在下发补丁数据完成后等待补丁执行结果。

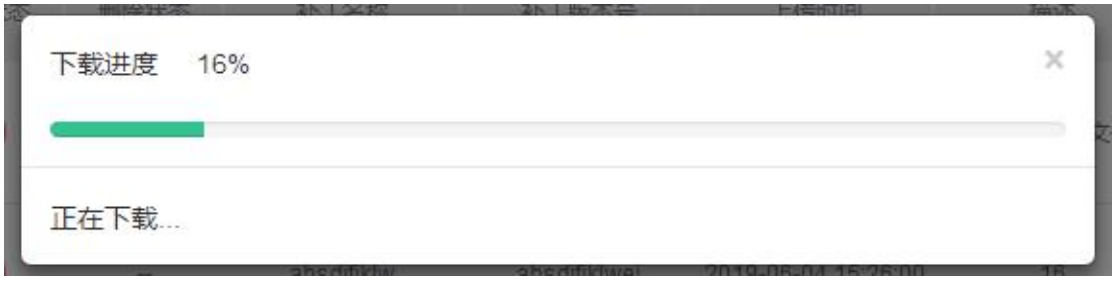
输入待验证设备的 MAC 地址，点击“立即验证”，弹出“下载进度”提示框，开始设备的验证升级流程。

1) 准备下载：等待设备请求下载该补丁



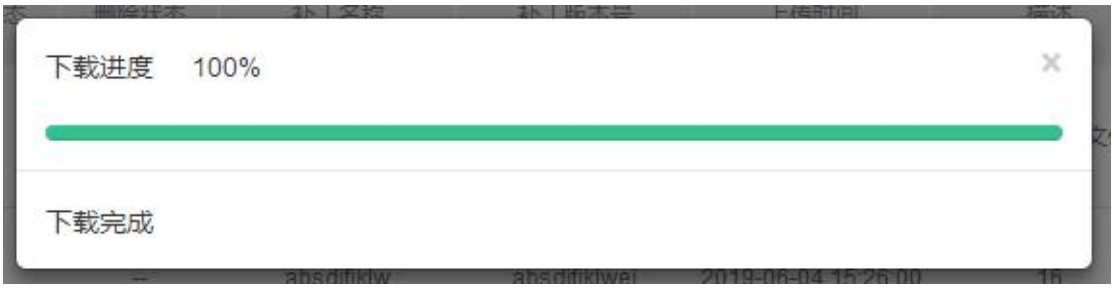

验证补丁时，云端会通知设备拉取补丁列表，当收到拉取列表的回调时，请调用《青莲云嵌入式 SDK 开发使用文档》5.2.2 节的函数，向云端请求下载该补丁。

2) 正在下载：可以看到补丁下载进度



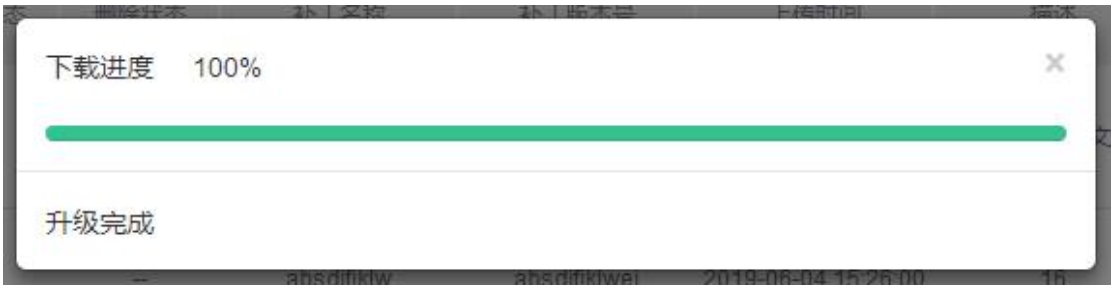

下载补丁时会调用《青莲云嵌入式 SDK 开发使用文档》5.3 节的函数，函数实现在文件 iot_interface.c 中，请根据具体平台自行编码实现。

3) 下载完成：等待补丁执行结果，等待时间 1 分钟




下载完成后，请运行补丁，并将补丁的执行结果通过《青莲云嵌入式 SDK 开发使用文档》5.2.4 节的函数，通知云端。

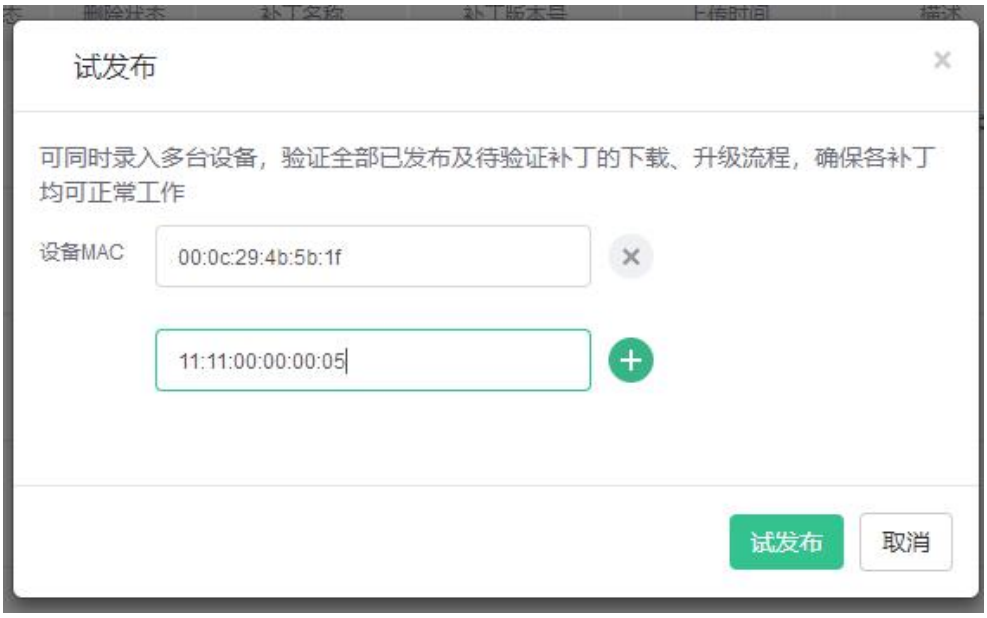
4) 升级完成：运行新固件成功上报给云端后，会提示"升级完成




## 3.3 试发布

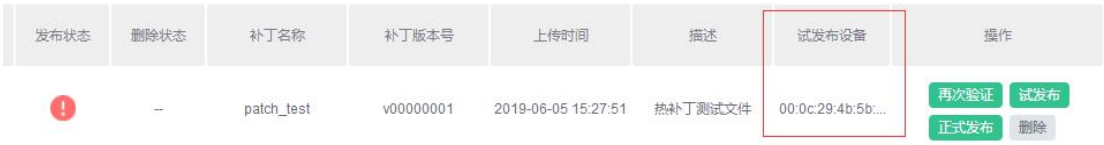
为保证在线设备下载所有未下载补丁后，能够正常工作，要先对整个升级流程进行验证，这个验证过程就是试发布的过程。

点击补丁版本右边的 试发布 按钮，弹出"试发布"对话框，如下图所示：




◆ 设备 mac：一次可选择一个或多个设备进行试发布。发布成功后，指定设备在每次拉取补丁列表的时候会拉到所有已发布版本和该设备的试发布版本。

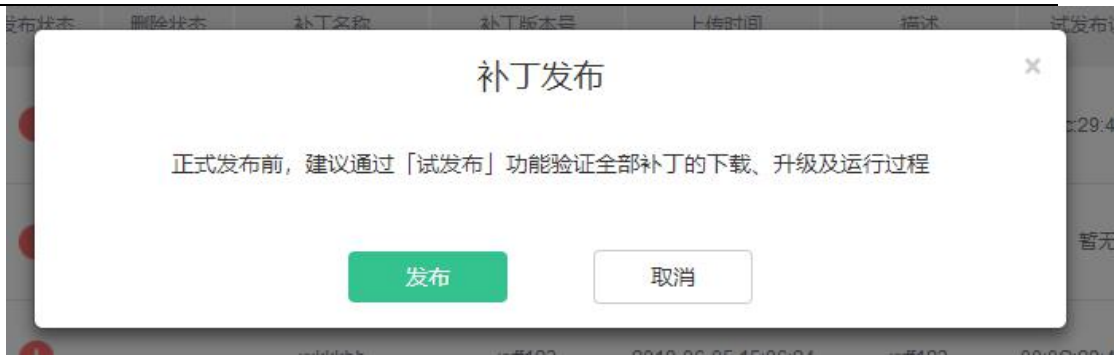
可在试发布设备栏查看当前补丁试发布设备的 mac，如下图所示：




当试发布验证后，可以对补丁进行正式发布操作。

## 3.4 正式发布

补丁验证/试发布成功后，点击 正式发布 按钮弹出如下"补丁发布"对话框，建议在发布前通过试发布功能验证全部补丁的下载，升级和运行过程。




点击发布后，补丁将正式发布。若某设备在验证/试发布过程中已下载该补丁，则云端不会重复向该设备推送这条补丁。

## 4 拉取补丁列表

设备上电后，首先检测当前固件版本：

1. 当前部件不是最新版本，等待 ota 升级重启。
2. 当前固件为最新版本，会立即向云端请求补丁列表。之后每六个小时向云端请求一次补丁列表。

补丁在不同状态下拉取到的列表不同，如下表所示：

| 补丁状态 | 获取补丁列表内容 |
| --- | --- |
| 验证补丁 | 当前验证补丁 |
| 试发布补丁 | 处于试发布状态的补丁 + 已发布补丁 |
| 发布补丁 | 该设备未下载的所有已发布补丁 |

云端发送的补丁列表会通过回调函数 iot_patchs_list_cb（）发送给用户。函数 iot_patchs_list_cb（）需要用户自己实现。

示例:

```c
void iot_patchs_list_cb( iot_s32 count, patchs_list_t* patchs )
{
    int i = 0;

    printf("patchs count:%d\r\n", count);
    g_patch_num = count;
    g_patchs_list = patchs;

    for(i = 0; i < count; i++)
    {
        printf("name:%s | version:%s | total_len:%d\r\n", g_patchs_list[i].name, g_patchs_list[i].version, g_patchs_list[i].total_len);
    }
}
```

## 5 请求下载

收到云端发来的列表后，用户根据返回的列表判断哪些补丁是需要下载的，依次调用 iot_patch_req()向云端请求下载。

若有多个补丁请求下载，需保证在上一个补丁已升级完成后再调 iot_patch_req()。

## 6 补丁下载

下载时，云端会将上传的固件文件分块下发，设备接收固件块的接口是 iot_chunk_cb（）。函数 iot_chunk_cb()需要用户实现。示例如下：

```c
iot_s32 iot_chunk_cb ( iot_u8 chunk_stat, iot_u32 chunk_offset, iot_u32 chunk_size, const iot_s8* chunk )
{
    printf("get %s chunk, offset = %d, size = %d\r\n", (IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_OTA) ? "OTA" : "PATCH",
                                                       chunk_offset, chunk_size);

    //第一个数据包，准备升级空间
    if ( chunk_offset == 0 )
    {
        //擦除flash，准备写入固件/补丁
        if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_OTA)
        {
            /* 固件示例
            * flash_erase(start_address);
            */
        }
        else if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_PATCH)
        {
            /* 补丁示例
            * fp = fopen("/root/iot_patch_0102", "wb");
            */
        }
    }

    //将固件/补丁数据块写入升级空间
    if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_OTA)
    {
        /* 固件示例
        * flash_write( chunk_offset, chunk, chunk_size );
        */
    }
    else if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_PATCH)
    {
        /* 补丁示例
        * fwrite(chunk, chunk_size, 1, fp);
        */
    }
```

```
    //最后一个数据包
    if ( IOT_CHUNK_IS_LAST(chunk_stat) )
    {
        if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_OTA)
        {
            /* 固件示例，一般需要校验整个固件
            * verify_firmware(start_address);
            */
        }
        else if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_PATCH)
        {
            /* 补丁示例，一般需执行补丁升级流程
            * fclose(fp);
            * fp = NULL;
            */
        }
        printf("get chunks over, wait to upgrade...\r\n");

        if(IOT_CHUNK_TYPE(chunk_stat) == IOT_CHUNK_PATCH)
        {
            /* 测试升级补丁
            * ret = test_patch("/root/iot_patch_0102");
            * 将补丁升级状态上报至云端
            * iot_patch_end("iot_patch", "0102", ret);
            */
        }
    }
    return ACK_OK;
}
```

## 7 补丁升级

下载完成后，设备端需检测运行该补丁，然后将执行结果通过 iot_patch_end（）返回给云端。