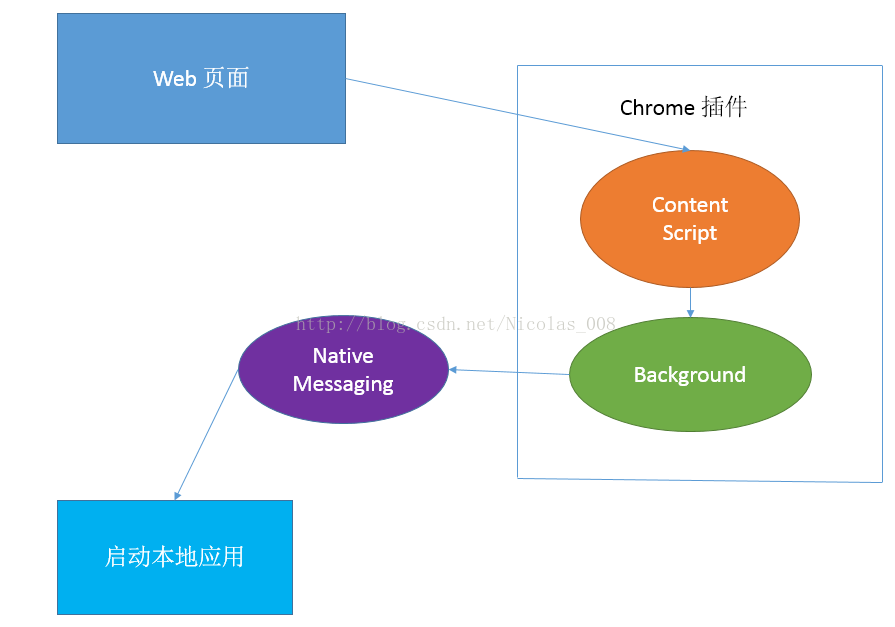
我们编写chrome插件，使用Native Messaging技术来和本地应用通信，来达到交互目的。

## Chrome插件交互图



## 编写 Chrome 插件实例：读取加密狗的信息

编写该实例需要2大模块：Chrome插件 、 本地Host主机程序

### 3.1编写Chrome 插件

该插件取名：LoadDogKey

在本地D盘（路径可变，我只是为了简化开发而已）新建文件夹“LoadDogKey”，里面新建三个文件：manifest.json(名字不可改,插件必须文件)，content.js(content script文件，负责与网站页面交互)，background.js(文件名可改, 后台文件)。

插件配置文件manifest.json：

*{*

*"name" : "LoadDogKey",*

*"version" : "1.0.1",*

*"description" : "load dog key info ",*

*"background" : {*

*"scripts": ["background.js"]*

*},*

*"permissions" : [*

*"nativeMessaging",*

*"tabs",*

*"http://\*/\*",*

*"https://\*/\*"*

*],*

*"content\_scripts": [*

*{*

*"matches": ["http://\*/\*","https://\*/\*"],*

*"js": ["content.js"],*

*"run\_at":"document\_start"*

*}*

*],*

*"minimum\_chrome\_version" : "6.0.0.0",*

*"manifest\_version": 2*

*}*

里面的内容是：插件名称、版本号、描述、后台运行脚本的JS文件、运行权限、内容脚本、最低Chrome运行版本、manifest版本。“permissions”中一定要有“nativeMessaging”。

插件代理文件(我是这么理解的) content.js

*document.addEventListener('myCustomEvent', function(evt) {*

*chrome.runtime.sendMessage({type: "loaddogkey", message: evt.detail}, function(response) {*

*console.log(response)*

*});*

*}, false);*

*chrome.runtime.onMessage.addListener(HandleMessage);*

*function HandleMessage(request, sender, sendResponse)*

*{*

*if (request.type == "loaddogkey\_callback")*

*{*

*document.getElementById('txtDogKey').value = request.dogKey;*

*document.getElementById('txtDogProtectValue').value = request.dogProtectValue;*

*document.getElementById('txtDogPublicValue').value = request.dogPublicValue;*

*}*

*};*

第一个函数：监听Web页面是否有自定义的请求事件，如果有，则后台文件发送消息请求，并且传递请求参数。

第二个函数：监听是否有后台文件处理的返回结果，来进行页面的相应更新。

后台文件background.js

*chrome.runtime.onMessage.addListener(*

*function(request, sender, sendResponse) {*

*if (request.type == "loaddogkey"){*

*connectToNativeHost(request.message);*

*}*

*return true;*

*}*

*);*

*function connectToNativeHost(msg)*

*{*

*chrome.runtime.sendNativeMessage('www.dcjet.com.loaddogkey',*

*{ text: msg},function(response) {*

*if (chrome.runtime.lastError) {*

*console.log("ERROR: " + chrome.runtime.lastError.message);*

*} else {*

*//console.log("Messaging host sais: ", response);*

*chrome.tabs.query({active: true}, function(tabs) {*

*chrome.tabs.sendMessage(tabs[0].id, {type:"loaddogkey\_callback",dogKey : response.dogKey,dogProtectValue : response.dogProtectValue ,dogPublicValue : response.dogPublicValue }, function(response) {*

*});*

*});*

*}*

*});*

*}*

第一个函数：插件的监听程序，通过参数“type”来判断，是否为我需要的消息请求，如果不是，进行忽略，如果是就处理。

第二个函数：链接本地主机程序，通过native messaging 来进行交互，通过chrome.tabs.sendMessage方法将主机程序的输出结果返回。其中'www.dcjet.com.loaddogkey'是本地主机的名称（并非exe的名称，可自定义，但必须与本地Host主机配置文件中名称相同）

### 3.2本地Host主机程序

在本地C盘（路径可变，我只是为了简化开发而已）新建文件夹“Native Plugins”，新建配置文件manifest.json

*{*

*"name": "www.dcjet.com.loaddogkey",*

*"description": "load dog key info.",*

*"path": "C:\\Native Plugins\\LoadDogKey.exe",*

*"type": "stdio",*

*"allowed\_origins": [*

*"chrome-extension://gefclbbonjckppkbcgalbjblcglnnhik/"*

*]*

*}*

里面内容是:主机名称，描述，主机程序的路径，type(这个必须是stdio,他只支持标准输入输出流),插件ID这个后面会得到。

还需要编写一个exe程序（可用C#或者C++编写）：

*int main(int argc, char\* argv[]) {*

*//读取加密狗的序列号*

*CString dogKey = ReadSerial("20083");*

*//读取加密狗的保护区*

*CString dogProtectValue = ReadProtectArea("20083","4294967294");*

*//读取加密狗的公共区*

*CString dogPublicValue = ReadPublicArea("20083","4294967294");*

*string message="{\"dogKey\": \""+dogKey+"\",\"dogProtectValue\": \""+dogProtectValue+"\",\"dogPublicValue\": \""+dogPublicValue+"\"}";*

*unsigned int len = message.length();*

*cout << char(((len>>0) & 0xFF))*

*<< char(((len>>8) & 0xFF))*

*<< char(((len>>16) & 0xFF))*

*<< char(((len>>24) & 0xFF));*

*cout << message <<endl;*

*return 0;*

*}*

*将生成的exe程序放到*manifest.json文件的目录中。

需要在客户机的注册表中注册主机程序

*reg add "HKEY\_CURRENT\_USER\Software\Google\Chrome\NativeMessagingHosts\www.dcjet.com.loaddogkey" /d "C:\Native Plugins\manifest.json" /t REG\_SZ /f*

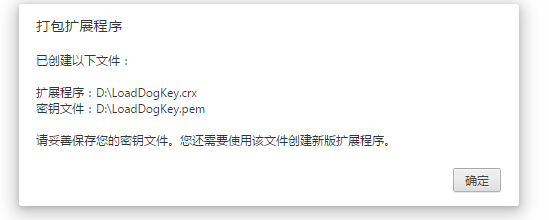
*或*

*reg add "HKEY\_LOCAL\_MACHINE\SOFTWARE\Google\Chrome\NativeMessagingHosts\www.dcjet.com.loaddogkey" /d "C:\Native Plugins\manifest.json" /t REG\_SZ /f*

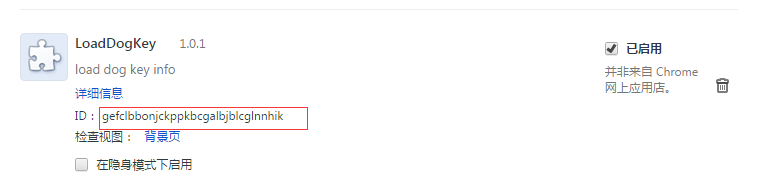
里面的路径就是我们manifest.json文件的路径。

现在我们插件的主要工作就编写完了

## 插件打包

打开我们的Chrome浏览器，输入chrome://extensions/,勾选“开发者模式”，点击“打包扩这程序”按钮，选择我们刚刚创建的插件文件夹（D:\LoadDogKey）,点击“打包扩展程序”，将生成扩展程序文件（LoadDogKey.crx）和秘钥文件（LoadDogKey.pem）。如图



将刚生成的crx文件拖动到我们的扩展中，在弹出的对话框中点击“添加”，可以查看生成的ID,并将该ID替换主机manifest.json文件中"chrome-extension”:对应的值

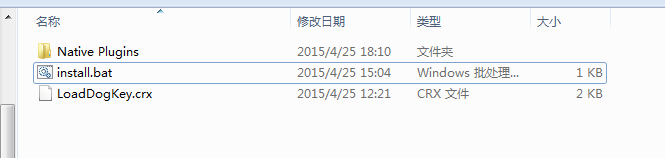
## 客户端下载安装

将刚我们的主机程序文件夹和生成的插件程序放在一个文件夹下，然后我写了个批处理方便客户安装，批处理文件install.bat如下：

*xcopy "Native Plugins" "C:\Native Plugins\" /y*

*reg add "HKEY\_CURRENT\_USER\Software\Google\Chrome\NativeMessagingHosts\www.dcjet.com.loaddogkey" /d "C:\Native Plugins\manifest.json" /t REG\_SZ /f*

*reg add "HKEY\_LOCAL\_MACHINE\SOFTWARE\Google\Chrome\NativeMessagingHosts\www.dcjet.com.loaddogkey" /d "C:\Native Plugins\manifest.json" /t REG\_SZ /f*

最终我们需要给客户下载的压缩文件如下所示：

客户可以通过在拿到这个文件夹后，只需要将我们的插件程序crx拖到Chrome扩展程序中，然后双击运行install.bat文件即可。

## 远程测试站点

新建一个远程测试页面和我们的插件交互

例如index.html （http://XXXXX/index.html）

*<!DOCTYPE HTML>*

*<html>*

*<head>*

*<script>*

*function loadDogKeyInfo(){*

*var evt = document.createEvent("CustomEvent");*

*evt.initCustomEvent('myCustomEvent', true, false, "dogID");*

*document.dispatchEvent(evt);*

*}*

*function load(){*

*document.getElementById('btnLoadDogKey').click();*

*}*

*</script>*

*</head>*

*<body onload="load()">*

*This is Test Page!!! Get Dog Key Info... <br/><br/><br/><br/><br/>*

*<button type="button" onClick="loadDogKeyInfo()" id="btnLoadDogKey" style="display:none;">Load Dog Key</button>*

*<br/><br/>*

*DogKey<input type="text" id="txtDogKey" style="width:200px;"/> <br/>*

*DogProtectValue<input type="text" id="txtDogProtectValue" style="width:200px;"/><br/>*

*DogPublicValue<input type="text" id="txtDogPublicValue" style="width:200px;"/><br/>*

*</body>*

*</html>*

通过发送事件和插件的本地代理（content\_script.js）脚本进行通信。

这样我们就可以很容易的通过插件来读取我们usdkey 中的信息了。

更详细的说明请参考官网：

<https://developer.chrome.com/extensions/messaging>

<https://developer.chrome.com/extensions/nativeMessaging>

<https://developer.chrome.com/extensions/content_scripts>