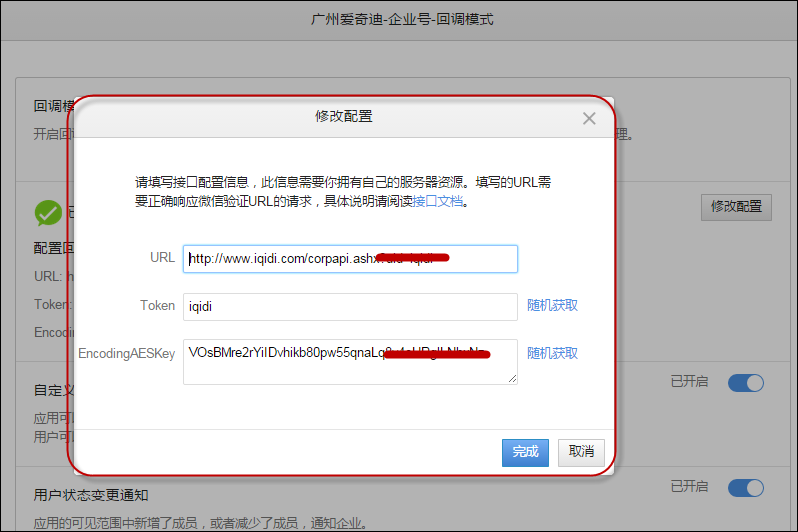
**[C#开发微信门户及应用(21)-微信企业号的消息和事件的接收处理及解密](http://www.cnblogs.com/wuhuacong/p/3996000.html)**

在上篇随笔《[C#开发微信门户及应用(19)-微信企业号的消息发送（文本、图片、文件、语音、视频、图文消息等）](http://www.cnblogs.com/wuhuacong/p/3995494.html)》介绍了有关企业号的消息发送，官方特别声明消息是不用加密发送的。但是在回调的服务器上，也就是我们网站的服务器上，微信传过来的消息是加密的，需要我们调用类库对消息和事件进行解密操作，由于官方的例子不全，因此摸索了不少时间，最终顺利解密收到的各种消息和事件。本文主要介绍 微信企业号的消息和事件的接收处理及解密操作。

**1、企业号回调模式的设置**

和公众号一样，微信企业号如果需要进行二次开发，也是需要在后台设置好对应的回调参数，如下界面所示。



设置好这些后，检查通过后，我们就可以在自己微信应用服务器上进行消息的收发操作了。

在回调的消息入口处，我们需要对POST数据和普通的GET数据进行分开处理，GET数据是微信自身的验证处理，POST数据是微信消息的交互操作。

[复制代码](javascript:void(0);)

/// <summary>

/// 企业号回调信息接口。统一接收并处理信息的入口。

/// </summary>

public class corpapi : IHttpHandler

{

/// <summary>

/// 处理企业号的信息

/// </summary>

/// <param name="context"></param>

public void ProcessRequest(HttpContext context)

{

[复制代码](javascript:void(0);)

上面我们定义了一个一般应用处理程序来对消息进行处理。

然后我们分开不同的消息类型（POST、GET 方式），针对性的进行处理。

[复制代码](javascript:void(0);)

if (HttpContext.Current.Request.HttpMethod.ToUpper() == "POST")

{

using (Stream stream = HttpContext.Current.Request.InputStream)

{

Byte[] postBytes = new Byte[stream.Length];

stream.Read(postBytes, 0, (Int32)stream.Length);

postString = Encoding.UTF8.GetString(postBytes);

}

if (!string.IsNullOrEmpty(postString))

{

Execute(postString, accountInfo);

}

}

else

{

Auth(accountInfo);

}

[复制代码](javascript:void(0);)

**2、微信回调消息的验证**

 下面是微信对于回调模式，验证URL的说明。

**验证URL有效性**

当你提交以上信息时，企业号将发送GET请求到填写的URL上，GET请求携带四个参数，**企业在获取时需要做urldecode处理**，否则会验证不成功。

|  |  |  |
| --- | --- | --- |
| **参数** | **描述** | **是否必带** |
| msg\_signature | 微信加密签名，msg\_signature结合了企业填写的token、请求中的timestamp、nonce参数、加密的消息体 | 是 |
| timestamp | 时间戳 | 是 |
| nonce | 随机数 | 是 |
| echostr | 加密的随机字符串，以msg\_encrypt格式提供。需要解密并返回echostr明文，解密后有random、msg\_len、msg、$CorpID四个字段，其中msg即为echostr明文 | 首次校验时必带 |

企业通过参数msg\_signature对请求进行校验，如果确认此次GET请求来自企业号，那么**企业应用对echostr参数解密并原样返回echostr明文(不能加引号)**，则接入验证生效，回调模式才能开启。

后续回调企业时都会在请求URL中带上以上参数（echostr除外），校验方式与首次验证URL一致。

根据上面的说明，我们需要获取这些参数，然后调用微信提供的消息处理函数进行加解密处理。

在验证URL的Auth(accountInfo);操作里面，我们可以看到核心的内容如下所示，就是获取到这些传递过来的参数信息，然后交给基类处理消息的签名内容。

[复制代码](javascript:void(0);)

#region 具体处理逻辑

string echoString = HttpContext.Current.Request.QueryString["echoStr"];

string signature = HttpContext.Current.Request.QueryString["msg\_signature"];//企业号的 msg\_signature

string timestamp = HttpContext.Current.Request.QueryString["timestamp"];

string nonce = HttpContext.Current.Request.QueryString["nonce"];

string decryptEchoString = "";

if (new CorpBasicApi().CheckSignature(token, signature, timestamp, nonce, corpId, encodingAESKey, echoString, ref decryptEchoString))

{

if (!string.IsNullOrEmpty(decryptEchoString))

{

HttpContext.Current.Response.Write(decryptEchoString);

HttpContext.Current.Response.End();

}

}

#endregion

[复制代码](javascript:void(0);)

验证代码部门如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// 验证企业号签名

/// </summary>

/// <param name="token">企业号配置的Token</param>

/// <param name="signature">签名内容</param>

/// <param name="timestamp">时间戳</param>

/// <param name="nonce">nonce参数</param>

/// <param name="corpId">企业号ID标识</param>

/// <param name="encodingAESKey">加密键</param>

/// <param name="echostr">内容字符串</param>

/// <param name="retEchostr">返回的字符串</param>

/// <returns></returns>

public bool CheckSignature(string token, string signature, string timestamp, string nonce, string corpId, string encodingAESKey, string echostr, ref string retEchostr)

{

WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(token, encodingAESKey, corpId);

int result = wxcpt.VerifyURL(signature, timestamp, nonce, echostr, ref retEchostr);

if (result != 0)

{

LogTextHelper.Error("ERR: VerifyURL fail, ret: " + result);

return false;

}

return true;

}

[复制代码](javascript:void(0);)

**3、企业号的消息处理**

 上面介绍了，微信企业号对URL的验证过程，还有另外一个消息处理过程，就是微信服务器把消息发送给我们自己的应用服务器进行处理的过程，我们应用服务器需要在收到消息后，及时进行常规回复处理。

也就是下面的代码逻辑。

[复制代码](javascript:void(0);)

if (HttpContext.Current.Request.HttpMethod.ToUpper() == "POST")

{

using (Stream stream = HttpContext.Current.Request.InputStream)

{

Byte[] postBytes = new Byte[stream.Length];

stream.Read(postBytes, 0, (Int32)stream.Length);

postString = Encoding.UTF8.GetString(postBytes);

}

if (!string.IsNullOrEmpty(postString))

{

Execute(postString, accountInfo);

}

}

[复制代码](javascript:void(0);)

同样，我们给微信服务器回应消息的时候，我们也需要获得相应的参数，然后再行构造信息回答。

string echoString = HttpContext.Current.Request.QueryString["echoStr"];

string signature = HttpContext.Current.Request.QueryString["msg\_signature"];//企业号的 msg\_signature

string timestamp = HttpContext.Current.Request.QueryString["timestamp"];

string nonce = HttpContext.Current.Request.QueryString["nonce"];

而另外一些参数信息，则是来源于我们企业号账号的配置参数。

//获取配置参数并对加解密函数初始化

string CorpToken = accountInfo.Token;

string AESKey = accountInfo.EncodingAESKey;

string CorpId = accountInfo.CorpID;

然后使用微信提供的消息加解密类，就可以顺利对消息进行加解密的处理了。具体操作代码如下所示。

[复制代码](javascript:void(0);)

//根据参数信息，初始化微信对应的消息加密解密类

WXBizMsgCrypt wxcpt = new WXBizMsgCrypt(CorpToken, AESKey, CorpId);

//对收到的密文进行解析处理

string sMsg = ""; // 解析之后的明文

int flag = wxcpt.DecryptMsg(signature, timestamp, nonce, postStr, ref sMsg);

if (flag == 0)

{

//LogTextHelper.Info("记录解密后的数据:");

//LogTextHelper.Info(sMsg);//记录解密后的数据

CorpApiDispatch dispatch = new CorpApiDispatch();

string responseContent = dispatch.Execute(sMsg);

//加密后并发送

//LogTextHelper.Info(responseContent);

string encryptResponse = "";

timestamp = DateTime.Now.DateTimeToInt().ToString();

wxcpt.EncryptMsg(responseContent, timestamp, nonce, ref encryptResponse, ref signature);

HttpContext.Current.Response.ContentEncoding = Encoding.UTF8;

HttpContext.Current.Response.Write(encryptResponse);

}

else

{

LogTextHelper.Info("解密消息失败！");

}

[复制代码](javascript:void(0);)

最终，我们把解密完成的消息交给对应的封装类进行统一处理就可以了。

CorpApiDispatch dispatch = new CorpApiDispatch();

string responseContent = dispatch.Execute(sMsg);

这样我们在企业号API的封装，就可以只需要关注消息如何应答的逻辑就可以了，其他的不用关心。

