PayNowRefundAPI_G

V1.6 (2022/11/17)

正式：https://www.paynow.com.tw/service/PayNowAPI_JS.aspx

測試：https://test.paynow.com.tw/service/ PayNowAPI_JS.aspx

編碼格式：所有參數傳遞時，請以 URL Encode 編碼，所有網頁字集為 UTF-8

操作流程：

退款 API 的使用流程

1.商家拋送 OP=GP 及 JStr 取得 PassCode 及 CheckNum; JStr 的內容為 json 格式並以 aes256 加密做 urlencoden 送出

2.paynow 以 aes256 解密取得 JStr 的內容，回吐 PassCode 及 CheckNum；組成 Json 格式後以 aes256 加密並做 urlencode 後回傳

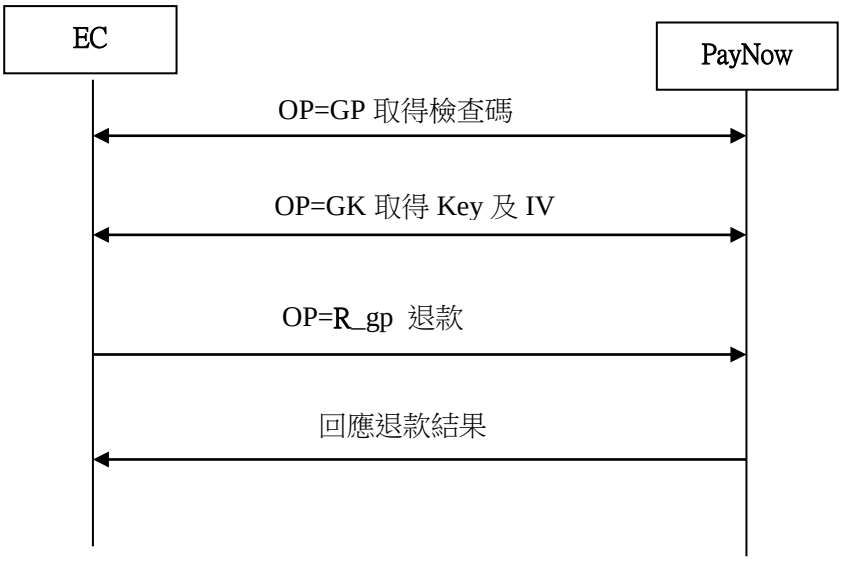
3.商家接收回應字串做 urldecode 後以 aes256 解密，解析 json 格式取得 CheckNum 並驗證 PassCode

4.商家以步驟 3 取得的 CheckNum 呼叫 OP=GK 的 API，JStr 的內容為 json 格式並以 aes256 加密做 urlencoden 送出

5.paynow 以 aes256 解密取得 JStr 的內容，回吐 PassCode、EncryptionKey 及 EncryptionIV；組成 Json 格式後以 aes256 加密並做 urlencode 後回傳

6.商家呼叫 OP= R_gp 的 API，以文件描述需要的參數組成 JsonString 後，利用步驟 5 取得的 EncryptionKey 及 EncryptionIV 將 JsonString 以 AES256 加密後，拆分成 JStr1 及 JStr2 並做 urlencode，拋送文件描述的參數共 6 個

7.paynow 回應退款結果



1. 檢核碼串接

傳遞方式：Http POST method UrlEncode

參數名稱	描述	型態	必須	備註
傳送 (Object)				
OP	判斷代號	string	Y	OP=GP；取得檢查碼 OP=GK；取得 Key 及 IV
JStr	檢核驗證資訊	string	Y	將下列參數組成 Json 格式字串，以 AES256 加密傳遞 (註 1) GP 取得隨機檢查碼： mem_cid：商家帳號(統編/身分證) PassCode：交易驗證碼 PassCode 產生方式請見(註 3) (註 8) TimeStr：時間戳 時間戳產生方式請見(註 4) GK 取得 Key 及 IV： mem_cid：商家帳號(統編/身分證) PassCode：交易驗證碼 PassCode 產生方式請見(註 3)(註 8) TimeStr：時間戳 時間戳產生方式請見(註 4) CheckNum：檢核碼 (8 碼)
回應 (Object)				
	WS 回應	string		Json 字串以 AES256 加密回傳，請以 AES256 解密 (註 2)
	Json 解密後 Object 內容			
	GP：			
	mem_cid	string		商家帳號
	PassCode	string		回覆驗證碼 (註 3)
	TimeStr	string		時間戳
	CheckNum	string		隨機檢核碼 (8 碼)
	GK：			
	PassCode	string		回覆驗證碼 (註 3)
	EncryptionKey	string		加密 Key
	EncryptionIV	string		加密 IV

檢核碼使用的 AES256 加密(註 1 及註 2)key 及 IV 固定為下：

Key：paynowencryptpaynowcomtw28229955
IV：encrypt282299550

1.1 檢核碼串接範例

```
string OP = "";
string timestr = StrRight(DateTime.Now.Year.ToString(), 1) + StrRight("000" + DateTime.Now.DayOfYear.ToString(), 3) + StrRight("0" + DateTime.Now.Hour.ToString(),
2) + StrRight("0" + DateTime.Now.Minute.ToString(), 2) + StrRight("0" + DateTime.Now.Second.ToString(), 2);
string GetPowerCheckStr = Mod_ChkObject.GetPowerCheck(mem_cid, timestr, true);
string GPassCode = Mod_ChkObject.GetPassCode(mem, GetPowerCheckStr, false, "");
string GPassCode = Mod_ChkObject.GetPassCode("28229955", GetCheckNumStr, false, "");
                    Obj_Encrypt.Obj_InPutGetPass ObjInPutGetPass = new Obj_Encrypt.Obj_InPutGetPass();
                    Obj_Encrypt.Obj_OutPutGetPass ObjOutPutGetPass = new Obj_Encrypt.Obj_OutPutGetPass();
                    Obj_Encrypt.Obj_OutPutEncryption ObjOutPutKey = new Obj_Encrypt.Obj_OutPutEncryption();
                    ObjInPutGetPass.mem_cid = "28229955";
                    ObjInPutGetPass.PassCode = GPassCode;
                    ObjInPutGetPass.TimeStr = timestr;
string JSS = JsonConvert.SerializeObject(ObjInPutGetPass);
string Jstr = Mod_ChkObject.AES256_Encrypt(JSS, "paynowencryptpaynowcomtw28229955", "encrypt282299550");
OP = "GP";
returnStr = SubOP(OP, Jstr, "", "", "", "", "", posturl);
returnStr = Mod_ChkObject.AES256_Decrypt(Server.UrlDecode(returnStr), "paynowencryptpaynowcomtw28229955", "encrypt282299550");
ObjOutPutGetPass = JsonConvert.DeserializeObject<Obj_Encrypt.Obj_OutPutGetPass>(Server.UrlDecode(returnStr));

GetPowerCheckStr = Mod_ChkObject.GetPowerCheck ("28229955", timestr, false);
GPassCode = Mod_ChkObject.GetPassCode("28229955", GetPowerCheckStr, true, ObjOutPutGetPass.CheckNum);
ObjOutPutGetPass.PassCode = GPassCode;
JSS = JsonConvert.SerializeObject(ObjOutPutGetPass);
Jstr = Mod_ChkObject.AES256_Encrypt(JSS, "paynowencryptpaynowcomtw28229955", "encrypt282299550");

OP = "GK";
returnStr = SubOP(OP, Jstr, "", "", "", "", "", posturl);
returnStr = Mod_ChkObject.AES256_Decrypt(Server.UrlDecode(returnStr), "paynowencryptpaynowcomtw28229955", "encrypt282299550");
ObjOutPutKey = JsonConvert.DeserializeObject<Obj_Encrypt.Obj_OutPutEncryption>(Server.UrlDecode(returnStr));
```

分別取得如下內容(以下內容因 TimeStr 效期而有所差別)：

GP：

{"mem_cid":"28229955","PassCode":"CCE089C41567EFB631A3E82AA20D54B3F3D1BE841806C748AA9E39B57F301D73","TimeStr":"2321163000","CheckNum":"65813612"}

GK：

{"PassCode":"D35792712EBE651B297B4CD543086D47A68CCBB1338F19B19AD0EE8AA49F1355","EncryptionKey":"9a704b9059f14ea18103ac874a8d42c3","EncryptionIV":"adb710074b47cfc6"}

```
private string SubOP(string OP, string JStr, string JStr1, string JStr2, string mem_cid, string TimeStr, string CheckNum, string posturl)
    {
        string postdata = "";
        string returnStr = "";
        try
        {
            System.Text.UTF8Encoding encode = new System.Text.UTF8Encoding();
            byte[] encodedBytes;
            postdata = "OP=" + OP + "&JStr=" + Server.UrlEncode(JStr) + "&JStr1=" + Server.UrlEncode(JStr1) + "&JStr2=" + Server.UrlEncode(JStr2) + "&mem_cid="
+ mem_cid + "&TimeStr=" + TimeStr + "&CheckNum=" + CheckNum;
            encodedBytes = encode.GetBytes(postdata);
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            System.Net.HttpWebRequest HttpWReq = (System.Net.HttpWebRequest)System.Net.WebRequest.Create(posturl);
            HttpWReq.Method = "POST";
            HttpWReq.KeepAlive = true;
            HttpWReq.ContentType = "application/x-www-form-urlencoded";
            HttpWReq.ContentLength = encodedBytes.Length;
            HttpWReq.AllowAutoRedirect = true;
            HttpWReq.UserAgent = "PayNowapiJS";
            using (System.IO.Stream newStream = HttpWReq.GetRequestStream())
            {
                newStream.Write(encodedBytes, 0, encodedBytes.Length);
                using (System.Net.HttpWebResponse HttpResp = (System.Net.HttpWebResponse)HttpWReq.GetResponse())
                {
                    using (System.IO.StreamReader reader = new System.IO.StreamReader(HttpResp.GetResponseStream()))
                    {
                        returnStr = reader.ReadToEnd();
                    }
                }
            }
        }
        catch (Exception ex)
        {
            returnStr = ex.ToString();
        }
        finally
        {
            string lognameRsp = DateTime.Now.ToString("yyyyMMdd") + "_testGP_Resp_log.txt";
            WriteLog(lognameRsp, "Response_End:" + postdata + "_returnStr:" + returnStr);
        }
        return returnStr;
    }
```

2. 新增退款

傳遞方式：Http POST method UrlEncode

參數名稱	描述	型態	長度	必須	備註
OP	**服務代號**	**string**		**Y**	**OP = R_gp**
			傳送		
JStr1/JStr2	**Json 字串**	**string**		**Y**	請將下列參數組成 Json 格式字串 **mem_type：商家類型 (數字 1.買家 2.賣家)** **buysafeno：PayNow 訂單編號** **mem_cid：商家帳號** **passcode：交易驗證碼** **請以下列組合成字串後使用 SHA-1 雜湊函數加密並轉大寫** **mem_cidordernorefundprice 商家交易密碼** **mem_bankaccno：退款入帳帳號** **accountbankno：退款入帳銀行代碼** **mem_bankaccount：退款入帳銀行名稱** **refundvalue：退款原因** **refundmode：退款模式(1 發起退款 2 確認退款 3 取消退款(消費者))** **buyerid：消費者帳號 (若與原始交易不同請帶入)** **buyername：消費者姓名(若與原始交易不同請帶入)** **buyeremail：消費者 Email (若與原始交易不同請帶入)** **refundprice：退款金額** 將上列參數組成 Json 字串後 ,以檢核碼串接取得的 Key 及 IV(註 3)做 AES256 加密(註 1)後 將加密後字串對分拆解成 JStr1 及 JStr2 兩個參數，再做 UrlEncode 上傳
mem_cid	商家帳號	string	8-10	Y	商家帳號(統編/身分證)
TimeStr	時間戳	string	10	Y	加密傳送字串時使用的 TimeStr
CheckNum	隨機檢核碼	string	8	Y	加密傳送字串取得隨機檢查碼時回傳的 CheckNum
Json 格式字串參數內容					
mem_type	發起退款方	string	1	Y	數字轉字串 (EC 固定帶 2) 1：發起方為買家 2：發起方為賣家
buysafeno	PayNow 訂單編號	string	19	Y	
mem_cid	商家帳號	string	8-10	Y	法人請帶統編，個人請帶身分證
passcode	交易驗證碼	string		Y	請以下列組合成字串後使用 SHA-1 雜湊函數加密並轉大寫 mem_cidordernorefundprice 商家交易密碼
mem_bankaccno	退款入帳帳號	string	14-20	N	空字串；約定商家請帶入退款帳號

accountbankno	退款銀行代碼	string	7	N	空字串;約定商家請帶入退款銀行代碼共 7 碼(銀行代號+分行代號)
mem_bankaccount	退款銀行名稱	string		N	空字串；約定商家請帶入退款銀行名稱
refundvalue	退款原因	string	20	Y	退款原因
refundmode	發起方退款類型	string	1	Y	數字轉字串 (EC 固定帶 1) 1：發起退款 2：同意退款 3：取消退款(mem_type = 1 時可帶入)
buyerid	消費者帳號	string		N	系統預設抓取原始交易資料； 若非信用卡交易且與原交易不同請帶入(身分證-銀行檢核)
buyername	消費者姓名	string		N	系統預設抓取原始交易資料； 若非信用卡交易且與原交易不同請帶入(須為退款本人-銀行檢核)
buyeremail	消費者 Email	string		N	系統預設抓取原始交易資料； 系統將發送退款確認信件，請確實填寫
refundprice	退款金額	string		Y	數字轉字串 原訂單全額或部分金額 (當日交易或於信用卡收單請款完成前不得申請部分退款)
回應					
	回應字串	**string**			**成功：S_成功資訊,passcode** **失敗：F_錯誤訊息**

2.1 新增退款範例

Ex：C# code

```
{
        string JS1, JS2, Jstr, JSS, OP, paypasscode;
        Obj_Refund_JS objRefund = new Obj_Refund_JS();
        objRefund.mem_cid = "28229955";
        objRefund.buysafeno = "8000002211014651125";
        objRefund.mem_type = "2";
        objRefund.refundmode = "1";
        objRefund.refundprice = "1000";
        objRefund.refundvalue = "測試退款";
        paypasscode = objRefund.mem_cid + "202211010002" + System.Convert.ToString(Int(objRefund.refundprice)) +  商家交易密碼;
        paypasscode = FormsAuthentication.HashPasswordForStoringInConfigFile(paypasscode, "sha1");
        objRefund.passcode = paypasscode;

        OP = "R_gp";
        JSS = JsonConvert.SerializeObject(objRefund);
        Jstr = Mod_ChkObject.AES256_Encrypt(JSS, ObjOutPutKey.EncryptionKey, ObjOutPutKey.EncryptionIV);
        JS1 = Jstr.Substring(0, Int(Jstr.Length / 2));
        JS2 = Jstr.Substring(Int(Jstr.Length / 2), Jstr.Length - Int(Jstr.Length / 2));
        returnStr = SubOP(OP, "", JS1, JS2, objRefund.mem_cid, timestr, ObjOutPutGetPass.CheckNum, posturl); }
```

註 1、AES256 加密：

說明：下列範例使用 C#.net 編寫，僅供參考使用
此系統傳遞之AES256加解密，以檢核碼串接取得的Key值(**EncryptionKey**)及IV值(**EncryptionIV**)做加解密
Content：需要加密的內容
Keu：**EncryptionKey**
IV：**EncryptionIV**

Ex：C# code

```
public string AES256_Encrypt(string Content, string key, string IV)
        {
            byte[] ByteString = Encoding.UTF8.GetBytes(Content);
            byte[] ByteKeyString = Encoding.UTF8.GetBytes(key);
            byte[] ByteIVString = Encoding.UTF8.GetBytes(IV);
            RijndaelManaged RDEL = new RijndaelManaged();
            RDEL.Key = ByteKeyString;
            RDEL.IV = ByteIVString;
            RDEL.Mode = CipherMode.CBC;
            RDEL.Padding = PaddingMode.Zeros;
            ICryptoTransform cTransform = RDEL.CreateEncryptor();

            byte[] ResultByte = null;
            ResultByte = cTransform.TransformFinalBlock(ByteString, 0, ByteString.Length);
            string ReturnStr = Convert.ToBase64String(ResultByte, 0, ResultByte.Length);
            return ReturnStr;
        }
```

註 2、AES256 解密：

說明：下列範例使用 C#.net 編寫，僅供參考使用
此系統傳遞之AES256加解密，以檢核碼串接取得的Key值(**EncryptionKey**)及IV值(**EncryptionIV**)做加解密
Content：需要加密的內容
Keu：**EncryptionKey**
IV：**EncryptionIV**

Ex：C# code

```
public string AES256_Decrypt(string Content, string key, string IV)
        {
            byte[] ByteString = Convert.FromBase64String(Content);
            byte[] ByteKeyString = Encoding.UTF8.GetBytes(key);
            byte[] ByteIVString = Encoding.UTF8.GetBytes(IV);
            RijndaelManaged RDEL = new RijndaelManaged();
            RDEL.Key = ByteKeyString;
            RDEL.IV = ByteIVString;
            RDEL.Mode = CipherMode.CBC;
            RDEL.Padding = PaddingMode.Zeros;
            ICryptoTransform cTransform = RDEL.CreateDecryptor();

            byte[] ResultByte = null;
            ResultByte = cTransform.TransformFinalBlock(ByteString, 0, ByteString.Length);
            string ReturnStr = Encoding.UTF8.GetString(ResultByte);
            return ReturnStr;
```

註3、 PassCode產生規則(PassCode C#範例請見註8)：

GP 取得隨機檢查碼

以 mem_cid 及 GP規則加權檢核碼 (註5) 串接後做 SHA256雜湊加密字串(註6)後轉大寫帶入

回覆時的PassCode規則：

以 mem_cid 及 **GK規則**加權檢核碼 (註5) 串接後，以隨機檢查碼(CheckNum)為加密鑰 做 SHA256雜湊加密字串(註7)後轉大寫帶入

GK 取得Key及IV

以 mem_cid 及 GK規則加權檢核碼 (註5) 串接後，以隨機檢查碼(CheckNum)為加密鑰 做 SHA256雜湊加密字串(註7)後轉大寫帶入

回覆時的PassCode規則：

以 mem_cid 及 **GP規則**加權檢核碼 (註5) 串接後做 SHA256雜湊加密字串(註6)後轉大寫帶入

註 4、 TimeStr 時間戳格式：

Ex：以日期 2019-11-24 00:50:18 產生 10 碼數字

第 1 至 4 碼：太陽日，第 1 碼取西元年最後一位 2019 = 9

2 至 4 碼為一整年的起算天數，11/24 為今年的 328 天，所以 2 至 4 碼 = 328

第 5 至 10 碼：5-6 為小時數字 = 00；7-8 為分鐘數字 = 50；9-10 為秒分數字 = 18

1 至 10 碼組成時間戳 TimeStr = 9328005018

註 5、加權檢核碼產生規則：

分別以加權權數(23 碼)與加權基數(23 碼)各個數字相乘之後取個位數字，再取得**以 10 減去個數字相加後除以 10 得餘數後的數字**獲得加權權數檢查碼 1 碼 (10-餘數)，若為 10 則取 0；之後將加權權數前 15 碼字串與加權權數檢查碼 1 碼串接取得 16 碼數字；GPZ 與 GKZ 模式取得規則如下：

GP：取得隨機檢查碼

EX：商家帳號(mem_cid) = 028229955，TimeStr = 9328005018

加權基數(23 碼)：93193193193193193193193

加權權數(23 碼) :商家帳號 9 碼 ,不足左補 0 ;取商家帳號**前 5 碼(5 碼)** = 02822 + TimeStr(10 碼) = 9328005018

+ TimeStr 前 4 碼(4 碼) = 9328 + 商家帳號**後 4 碼**(4 碼) = 9955，加權權數(23 碼) = 02822932800501893289955

將權數與基數相乘取其個位數字相加

0	2	8	2	2	9	3	2	8	0	0	5	0	1	8	9	3	2	8	9	9	5	5
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
9	3	1	9	3	1	9	3	1	9	3	1	9	3	1	9	3	1	9	3	1	9	3
=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
0	6	8	18	6	9	27	6	8	0	0	5	0	3	8	81	9	2	72	27	9	45	15

取個數分別得到並相加：

0 + 6 + 8 + 8 + 6 + 9 + 7 + 6 + 8 +0 + 0 + 5 + 0 + 3 + 8 + 1 + 9 + 2 + 2 + 7 + 9 + 5 + 5 = 114

114 / 10 得餘數 4 ：加權檢查碼 取 10 – 4 = 6 加權檢核碼 = 0282293280050186

※組成回覆 PassCode 時的加權檢核碼會以 GK 的規則替換

GK：取得 Key 及 IV

EX：商家帳號(mem_cid) = 028229955，TimeStr = 9328005018

加權基數(23 碼)：93193193193193193193193

加權權數(23 碼) :商家帳號 9 碼，不足左補 0 ;取商家帳號**後 5 碼**(5 碼) = 29955 + TimeStr(10 碼) = 9328005018 + TimeStr 前 4 碼(4 碼) = 9328 + 商家帳號**前 4 碼(4 碼)** = 0282，加權權數(23 碼) = 29955932800501893280282

將權數與基數相乘取其個位數字相加

2	9	9	5	5	9	3	2	8	0	0	5	0	1	8	9	3	2	8	0	2	8	2
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
9	3	1	9	3	1	9	3	1	9	3	1	9	3	1	9	3	1	9	3	1	9	3
=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
18	27	9	45	15	9	27	6	8	0	0	5	0	3	8	81	9	2	72	0	2	72	6

取個數分別得到並相加：

8 + 7 + 9 + 5 + 5 + 9 + 7 + 6 + 8 +0 + 0 + 5 + 0 + 3 + 8 + 1 + 9 + 2 + 2 + 0 + 2 + 2 + 6 = 104

104 / 10 得餘數 4 ：加權檢查碼 取 10 – 4 = 6

加權檢核碼 = 2995593280050186

※組成回覆 PassCode 時的加權檢核碼會以 GP 的規則替換

註 6、SHA256 雜湊加密字串：

Ex：C# code

```
public string SHA256_Encrypt(string Content)
{
System.Security.Cryptography.SHA256CryptoServiceProvider sha256 = new System.Security.Cryptography.SHA256CryptoServiceProvider();
                byte[] ByteString = System.Text.Encoding.ASCII.GetBytes(Content);
                ByteString = sha256.ComputeHash(ByteString);
                string ReturnStr = null;
                foreach (byte bt in ByteString)
                    ReturnStr += bt.ToString("x2");
                return ReturnStr.ToUpper();
            }
```

註 7、SHA256 雜湊加密字串：

Ex：C# code

```
public string SHA256_HMACSHA256(string Content, string Key)
{
                byte[] ByteString = System.Text.Encoding.ASCII.GetBytes(Content);
                byte[] ByteKeyString = Encoding.ASCII.GetBytes(Key);
                byte[] HasBytes;
                HMACSHA256 sha256 = new HMACSHA256(ByteKeyString);
                HasBytes = sha256.ComputeHash(ByteString);
                string ReturnStr = BitConverter.ToString(HasBytes).Replace("-", "").ToUpper();
                return ReturnStr;
}
```

註 8. PassCode 範例

Ex：C# code

```
        public String GetPowerCheck(String mem_cid, String TimeStr, bool InPutFlag)
    {
        int CheckNum = 0;
        int intPN = 0;
        int intCN = 0;
        int intTime;
        string PowerNum = "93193193193193193193193";
        string BaseNum = "";
        string PowerCheck = "";
        string webno = mem_cid;
        try
        {
            //檢查傳入的時間長度是否10碼
            if (TimeStr.Length != 10) throw new Exception("TimeStr錯誤，請檢查長度是否正確");
            //檢查傳入的時間是否數字
            for (int i = 0; i < TimeStr.Length; i++) { if (Int32.TryParse(TimeStr.Substring(i, 1), out intTime) == false) throw new Exception("TimeStr錯誤，必須是
數字"); }
            //判斷身分證取右邊9碼
            if (Mod_ChkObject.Check_TW(mem_cid) == "0") { webno = StrRight(mem_cid, 9); }
            //判斷統編左補0
            if (Mod_ChkObject.ValidBID(mem_cid)) { webno = StrRight("0" + mem_cid, 9); }
            //是否GP
            if (InPutFlag == true)
            {
                BaseNum = StrLeft(webno, 5) + TimeStr + StrLeft(TimeStr, 4) + StrRight(webno, 4);
            }
            else
            {
                BaseNum = StrRight(webno, 5) + TimeStr + StrLeft(TimeStr, 4) + StrLeft(webno, 4);
            }

            if (BaseNum.Length != 23) throw new Exception("檢核碼錯誤，請檢查長度是否正確");
            for (int i = 0; i <= 22; i++)
            {
                intPN = Convert.ToInt16(BaseNum.Substring(i, 1));
                intCN = Convert.ToInt16(PowerNum.Substring(i, 1));
                CheckNum += Convert.ToInt16(StrRight(Convert.ToString(intPN * intCN), 1));
            }
            CheckNum = 10 - (CheckNum % 10);
            CheckString += StrRight("0" + Convert.ToString(CheckNum), 1);
```

```
            return PowerCheck;
        }
        catch (Exception ex)
        {
            return ex.Message.ToString();
        }

    public String GetPassCode(String mem_cid, String PowerCheck,bool ByKey,String key)
{
        string PassCode = "";
        try
        {
            PassCode = mem_cid + PowerCheck;
            if (ByKey == false) { PassCode = SHA256_Encrypt(PassCode); } else { PassCode = SHA256_HMACSHA256(PassCode,key); }
        }
        catch (Exception ex)
        {
            string strerr = "Error in HashCode : " + ex.Message;
            PassCode = strerr;
        }
        return PassCode;
}
```

附錄、退款服務錯誤代碼

R000	退款系統程式錯誤，請與系統商連絡
R001	申請人類別無填寫
R002	參數錯誤 (buysafeno) 訂單編號
R003	參數錯誤 (WebNo) 賣家會員帳號(公司統編或個人身分證)
R004	參數錯誤 (id) 買家會員帳號
R005	參數錯誤 (refundvalue) 退貨原因
R006	參數錯誤 (refunddate) 退貨日期
R007	參數錯誤 (ECPlatform) EC廠商名稱
R008	參數錯誤 (IDKey) IDKey檢驗識別碼
R009	參數錯誤 (PassCode) 檢驗識別碼
R010	參數錯誤 (refundmode) 退貨型態
R011	今日交易不可當日退款
R012	此交易賣家與申請退貨賣家不符
R013	查無此交易
R014	參數錯誤 (mem_bankaccno) 銀行帳號
R015	參數錯誤 (accountbankno) 銀行代碼
R016	參數錯誤 (mem_bankaccount) 撥款銀行分行名稱

R017	由賣家取消信用卡交易，系統自動通知發卡行退還額度
R018	非成功交易不得退款
R019	此貨品已配送不得退款
R020	此交易超過保管天數不得退款
R021	此交易小於30元不得退款
R022	此交易買家與申請退貨買家不符
R023	此交易已請款不得退款
R024	參數錯誤 (idRegist) 消費者身分證字號錯誤
R025	退貨取消錯誤，請聯絡系統商
R026	此買家無可退貨交易
R027	交易已報送發卡行待回覆，目前不得退款
R028	只能是商家發動退款
R029	只能是確認退款狀態
R030	只能是信用卡交易
R031	退款金額錯誤
R032	此交易不支援部分退款
R033	今日請款金額(+30)須大於退款金額。
R035	退款無法取消
R036	票券訂單已核銷，無法退款

R037	此為已撥款超過180天交易，退款作業完成須2-3個工作天；若於3個工作