综合业务平台 WebServices
接口使用说明

版本 V2.0

作者：	国政通	**日期：**	2012-8-1
修改:	国政通	**日期：**	2012-8-1
审批：		**日期：**	

目　录

一、 介绍

1. 简介

- 此接口适应于相关行业的用户对各数据进行查询，不需要人工干预，由用户 webservices 客户端系统自动处理。

2. 说明

- 使用此接口需要相应的用户名、密码等。

二、 服务器端说明

1. 简介

- JDK：java 5.0
- J2EE：1.4 以上
- 应用服务器：WebLogic 9.2.3
- WebServices 地址：http://gboss.id5.cn/services/QueryValidatorServices?wsdl
- 访问的用户名及密码：需另行提供给用户

2. 说明

- 用户名及密码、相关的权限等需要向国政通申请
- 客户需要提供静态 IP 地址或 IP 地址段(小于 10 个)

三、 客户端调用方法说明

1. 调用方法

- 描述：用户直接通过提供的 WebServices 地址，用户名和密码直接调用 ，由客户技术人员来开发客户端程序。
- 调用方法名：
  - 单条：querySingle
  - 批量: queryBatch
- 需要客户开发人员对 WebServices 非常了解。

2. 参数描述

- 批量和单条参数相同
- 1.用户名，2.密码，3.数据类型，4.输入参数
- 参数说明：
  - 用户名：单独提供
  - 密码：单独提供
  - 数据类型：

需要传值	数据类型涵义
1A020201	身份信息认证
1B010101	学历审核
1B020101	学籍审核

- 输入参数：
  - **单条**：每个参数据用,（英文下的逗号）分格，各个数据类型对应的举例如下。
    - 身份信息认证参数：姓名,身份证号，如“张三,510104197509202629”
    - 学历审核：姓名,身份证号，如“张三,510104197509202629”
    - 学籍审核：姓名,身份证号，如“张三,510104197509202629”
  - **批量**：数据与单条参数相同，多条数据之间有;(英文下的分号)隔开即可。此处不再做描述。

3. 返回格式说明

- 文件类型：Xml 格式
  - 节点说明：各数据类型返回的 xml 信息说明仅供参考，也可在调用后确定，以系统实际返回为准
- 各数据类型返回的 xml 信息举例
  - **单条**
    - **身份信息认证**
      - **一致**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
    <value>处理成功</value>
  </message>
  <policeCheckInfos>
    <policeCheckInfo name="刘静" id="110107197111011528">
      <message>
        <status>0</status>
        <value>查询成功</value>
      </message>
      <name desc="姓名">刘静</name>
      <identitycard desc="身份证号">110107197111011528</identitycard>
```

```
                <compStatus desc="比对状态">3</compStatus>
                <compResult desc="比对结果">一致</compResult>
                <checkPhoto desc="照片">照片(base64 码)</checkPhoto>
            </policeCheckInfo>
        </policeCheckInfos>
    </data>
```

- **不一致**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
    <value>处理成功</value>
  </message>
  <policeCheckInfos>
    <policeCheckInfo name="刘静静" id="110107197111011528">
      <message>
        <status>0</status>
        <value>查询成功</value>
      </message>
      <name desc="姓名">刘静静</name>
      <identitycard desc="身份证号">110107197111011528</identitycard>
      <compStatus desc="比对状态">2</compStatus>
      <compResult desc="比对结果">不一致</compResult>
    </policeCheckInfo>
  </policeCheckInfos>
</data>
```

- **库中无此号**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
    <value>处理成功</value>
  </message>
```

```
<policeCheckInfos>
  <policeCheckInfo name="李中华" id="532101196401260634">
    <message>
      <status>0</status>
      <value>查询成功</value>
    </message>
    <name desc="姓名">李中华</name>
    <identitycard desc="身份证号">532101196401260634</identitycard>
    <compStatus desc="比对状态">1</compStatus>
    <compResult  desc="比对结果">服务结果:库中无此号，请到户籍所在地进行核实！</compResult>
  </policeCheckInfo>
 </policeCheckInfos>
</data>
```

- **学历审核**

  - **查询成功-有数据**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
    <value>处理成功</value>
  </message>
  <eduInfos>
    <eduInfo name="钱江" id="330624198112100 45X">
      <message>
        <status>0</status>
        <value>查询成功</value>
      </message>
      <userName desc="姓名">钱江</userName>
      <identityCard desc="身份证号">330624198112100 45X</identityCard>
      <compResult desc="比对结果">成功返回</compResult>
      <graduate desc="毕业院校">浙江大学</graduate>
      <educationDegree desc="学历">本科</educationDegree>
      <enrolDate desc="入学年份">20040201</enrolDate>
```

```
        <specialityName desc="专业">英语</specialityName>
        <graduateTime desc="毕业时间">2006</graduateTime>
        <studyResult desc="毕业结论">毕业</studyResult>
        <studyStyle desc="学历类型">成人</studyStyle>
        <photo desc="照片">照片(base64 码)</photo>
      </eduInfo>
    </eduInfos>
</data>
```

- **查询成功-无数据**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
    <value>处理成功</value>
  </message>
  <eduInfos>
    <eduInfo name="钱江江" id="330624198112100 45X">
      <message>
        <status>1</status>
        <value>未查到数据</value>
      </message>
      <userName desc="姓名">钱江江</userName>
      <identityCard desc="身份证号">330624198112100 45X</identityCard>
      <compResult desc="比对结果">学历信息不存在</compResult>
    </eduInfo>
  </eduInfos>
</data>
```

◆ **学籍审核**

- **查询成功-有数据**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
```

```
    <value>处理成功</value>
  </message>
  <eduInfos>
    <eduInfo name="张雪" id="133001199004260280">
      <message>
        <status>0</status>
        <value>查询成功</value>
      </message>
      <userName desc="姓名">张雪</userName>
      <identityCard desc="身份证号">133001199004260280</identityCard>
      <compResult desc="比对结果">成功返回</compResult>
      <educationDegree desc="学习层次">本科</educationDegree>
      <graduate desc="学校名称">天津大学</graduate>
      <enrolDate desc="入学年份">2008</enrolDate>
      <specialityName desc="专业">化学工程与工艺</specialityName>
    </eduInfo>
  </eduInfos>
</data>
```

- **查询成功-无数据**

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>0</status>
    <value>处理成功</value>
  </message>
  <eduInfos>
    <eduInfo name="张雪雪" id="133001199004260280">
      <message>
        <status>1</status>
        <value>未查到数据</value>
      </message>
      <userName desc="姓名">张雪雪</userName>
      <identityCard desc="身份证号">133001199004260280</identityCard>
```

<compResult desc="比对结果">学籍信息不存在</compResult>

</eduInfo>

</eduInfos>

</data>

- **批量**
  - 返回的 xml 信息和单条保持一致，只是有多个 xml 标签信息。

  eg：身份信息认证　会有多个

  <policeCheckInfos>....<policeCheckInfos>

  <policeCheckInfos>....<policeCheckInfos>

◆当进行查询时，单条、批量返回的异常信息均相同，不做重复描述

异常信息举例：

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <message>
    <status>-9997</status>
    <value>你无权查询数据</value>
  </message>
</data>
```

➢ 节点描述

- 各数据源节点中的 message 中的 status 和 value 描述

message	status	value
	0	查询成功
	1	未查到数据
	2	查询失败

- 返回的节点 message 中的 status 和 value 的值如下

message	status	value
	0	处理成功
	-9999	参数数据不正确(部分参数为空)
	-9998	您的用户信息错误（用户名不存在）

	-9997	您无权查询数据
	-9996	参数请求数据过长
	-9995	该产品已暂停使用
	-9994	参数数据加密错误
	-990	系统异常
	9999	未查到数据
	-9919	参数数据不正确(参数格式不正确)
	-9929	参数数据不正确(参数个数不正确)
	-9917	您无权查询数据(ip 无权限)
	-9927	您无权查询数据(没有订购该产品)
	-9937	您无权查询数据(产品状态是开始状态)
	-9947	您无权查询数据(产品状态是暂停状态)
	-9957	您无权查询数据(产品状态是终止状态)
	-9967	您无权查询数据(测试量已经用完)
	-9977	您无权查询数据(账户余额不足)

四、 数据加密

1. 介绍

- 为了安全，接口在正式使用之后，将对传输的参数和返回的数据进行加密处理，即调用的所有参数都需要加密之后再传输，由我司系统解密后再进行处理，处理完成后，将完成的数据加密后返回给客户端，客户端对返回的数据进行解密处理，解密后的数据由用户客户端自行处理。

2. 算法

- 加密及解密均采用 DES 算法和 base64 编码相结合的方式对数据进行加解密。
- DES 算法均采用标准的方式获得，部分代码(java)如下：

加密算法：

```
public static String encode(String key,byte[] data) throws Exception{
    DESKeySpec dks = new DESKeySpec(key.getBytes());
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
    //key的长度不能够小于8位字节
    Key secretKey = keyFactory.generateSecret(dks);
    Cipher cipher = Cipher.getInstance(“DES/CBC/PKCS5Padding”);
    IvParameterSpeciv=new IvParameterSpec("12345678".getBytes());//向量
    AlgorithmParameterSpec paramSpec = iv;
```

```
        cipher.init(Cipher.ENCRYPT_MODE, secretKey,paramSpec);
        byte[] bytes = cipher.doFinal(data);
        return Base64.encode(bytes);
    }
```

解密算法：

```
public static byte[] decode(String key,byte[] data) throws Exception{
  try{
   SecureRandom sr = new SecureRandom();
   DESKeySpec dks = new DESKeySpec(key.getBytes());
   SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("DES");
   //key 的长度不能够小于 8 位字节
   Key secretKey = keyFactory.generateSecret(dks);
   Cipher cipher = Cipher.getInstance(ALGORITHM_DES);
   IvParameterSpec iv = new IvParameterSpec("12345678".getBytes());
   AlgorithmParameterSpec paramSpec = iv;
   cipher.init(Cipher.DECRYPT_MODE, secretKey,paramSpec);
   return cipher.doFinal(data);
  } catch (Exception e){
    e.printStackTrace();
    throw new Exception(e);
   }
 }
```

- 几组明码及加密后的数据，使用密钥 12345678,向量使用 12345678 进行加密及解密：

  明：abc ； 密：9YR6ZPdZufM=
  明：ABC ； 密：N3crakN1e3w=
  明：中国人 ； 密：qMeKyoDWvsE=
  明：在那遥远的地方 ； 密：6rtTnrF34mPkJ5SO3RiaaQ==

- 本系统均采用 DES 算法和 base64 编码相结合的方式进行加密解密，请客户技术人员根据我司提供的信息自行开发客户端及数据的加密解密处理。

五、 注意事项

1. IP 限制

- 为了安全，用户使用时，需提供您的 IP 地址，测试及上线均需提供。

2. 权限限制

- 限制每个用户调用接口数据的数量,防止非法调用；
- 批量查询（身份信息认证）的每批次不能超过 200 条，其它数据源每批次不能超过 1000。

六、 常见问题解答

1、 http://gboss.id5.cn/services/QueryValidatorServices?wsdl 该接口地址在 IE 浏览器不能访问，是什

么原因？

解答：没有开通 IP 权限。IP 权限分为测试和正式的 IP 权限。根据客户提供不同的 IP 地址具有不同的操作权限。

2、测试的账户信息(用户名和密码)和正式的账户信息是一样的吗？

解答：是一样的。一个客户只开通一个对应的账户信息。测试阶段和正式阶段只是该账户的两个阶段。

3、是否提供该 http://gboss.id5.cn/services/QueryValidatorServices?wsdl 接口的客户端调用的 demo？

解答：提供。Webservice 是一个通用的技术，任何语言(java、.net、c#、php 等)都支持，并且客户端程序可以自动生成。目前提供几个语言的 demo，详见下文“webservice 客户端调用 demo”.

4、连接 http://gboss.id5.cn/services/QueryValidatorServices?wsdl 该接口地址时报错，是什么原因？

解答：原因可能有下面的几个：

a)、没有 IP 权限。参见第一个问题的答案。

b)、webservice 客户端不是自动生成，是手动写的，可能存在问题，需要手动 debug 调试

5、调用时，webservice 服务端返回“您的用户信息错误”，是什么原因？

解答：原因可能有下面的几个：

a)、调用传的参数值没有进行 DES 加密。所有的参数值都需要加密。

b)、账户的用户名和密码为空或者不正确

6、采用 DES 加解密，DES 算法中 key 值和向量值是多少？客户可以自己指定吗？

解答：key 值和向量值默认都是 12345678。其中向量值是默认的，不能修改。key 的值可以指定。如果需要特殊指定，请事先告知。

7、在调用 webservice 服务端请求查询时，传的所有的参数值可以不进行 DES 算法加密吗？

解答：可以不进行加密。但是由于不进行加密，在传输的过程中，可能会被篡改，造成的损失将由客户自行承担。

8、调用成功后，webservice 服务端返回的 xml 信息是经过 DES 算法加密的吗？

解答：系统默认是加密方式的。如果抛出的参数没有加密并且已经设定好不加密方式，那么返回的 xml 不会加密；反之，则返回的 xml 会加密

9、是否提供 DES 算法和 base64 编码相结合的方式算法？

解答：提供。提供 java 版的算法。详细参加：“四：数据加密”。

10、调用时，webservice 服务端返回“您无权查询数据”，是什么原因？

解答：有以下几种原因：

a)、没有 IP 操作权限。客户绑定的 IP 地址范围不包含目前操作使用的 IP

b)、没有开通产品的查询权限

c)、账户余额不足

d)、测试使用量已经用完

e)、该产品没有订购

11、当客户测试成功后，想正式使用，怎么办？

解答：联系所属销售人员或技术支持人员，他们将为你开通正式使用。

12、调用成功后，返回的 xml 信息和该接口文档中相应产品的 xml 信息少，是什么原因？

解答：返回 xml 信息的多少，是所属销售定制的。请与所属销售联系。

13、在调用某些产品(比如：行驶证、驾驶证相关的产品)进行查询时，查询时间很长，是什么原因？

解答：有以上两个原因：

a)、在进行数据源方调用时，可能网络比较慢或者是数据库访问慢

b)、当调用数据源方查询数据失败后，系统会自动调用三次进行查询

14、进行相关产品查询时，得不到任何信息，是什么原因？

解答：有以上两个原因：

a)、webservice 客户端调用程序可能存在问题，请 debug 查询问题

b)、如果设置了客户端的请求响应时间，请尽量设置长一些

15、webservice 提供的接口文档中有很多相似的产品，这些产品有什么区别？

解答：文档中的所有产品的一个重要的区别是：返回的 xml 信息量不一样。

16、客户进行相关产品查询时，查询到的数据不是最新的，是什么原因？

解答：可能原因是数据源方的数据库还没有更新造成的

17、webservice 接口方式，不能进行页面的登陆，也不能进行数据查询量的统计，是什么原因？

解答：当客户使用 webservice 接口方式时，不提供页面的登陆和数据查询量的统计功能。但是如果客户需要，是可以提供这方面的支持，向所属销售申请权限即可。

七、 webservice 客户端调用 demo

1、JAVA 版

使用 https 协议调用核查接口需增加以下代码：

```
System.setProperty("javax.net.ssl.trustStore","CheckID.keystore");
```

CheckID.keystore：证书路径。

用 http://gboss.id5.cn/services/QueryValidatorServices?wsdl 自动生成 webservice 的代理对象。然后查询：

```
QueryValidatorServicesProxy proxy = new QueryValidatorServicesProxy();
proxy.setEndpoint("http://gboss.id5.cn/services/QueryValidatorServices?wsdl");
QueryValidatorServices service = proxy.getQueryValidatorServices();
String userName = "username";//用户名
String password = "password";//密码
System.setProperty("javax.net.ssl.trustStore", "CheckID.keystore");
String resultXML = "";
String datasource = "1A020201";//数据类型
//单条
String param = "刘丽萍,210204196501015829";//输入参数
```

```
resultXML = service.querySingle(userName, password, datasource, param);
//批量
String params = "王茜,150202198302101248;吴晨晨,36252519821201061x;王鹏,110108197412255477";
resultXML = service.queryBatch(userName, password, datasource, params);
```

2、.NET 版

使用 https 协议调用核查接口需要做到：

先在 IE 中访问 webservice 的 wsdl 地址，根据提示将证书加载到 IE，然后根据接口 wsdl 地址，用.net 提供的 wsdl.exe 生成 QueryValidatorServices 类。

在调用 webservice 接口前需加入一段代码：

System.Net.ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };

它的作用就是接受所有的证书，也就是在我们 SSL 中的流程中，检查证书 CA 是否受信这部分省略，就好比我们访问一些非受信证书的网站跳出的提示框我们点击确认一样。

部分代码如下：

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {

System.Net.ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
            QueryValidatorServices client = new QueryValidatorServices();
            string name = "李耀";
            try
            {
                //调用接口
                //单条
                string result = client.querySingle("username", "password", "1A020201", "王鹏,110108197412255477");//需要注意：所有的参数都需要加密
```

```
            //批量
            //string result = client.queryBatch("username", "password",
"1A020201", "王鹏,110108197412255477;吴晨,36252519821201061x");//需要注意：所有的参数都需要加密
            //解析 XML 字符串
            XmlDocument xd = new XmlDocument();
            xd.LoadXml(result);
            //获取根节点
            XmlNode xnRoot = xd.SelectSingleNode("CheckResult");
            string auth = string.Empty;
            string service = string.Empty;
            string photo = string.Empty;
            string photot = string.Empty;
            string verify_rlt = string.Empty;
            if (xnRoot != null)
            {
                auth = xnRoot.ChildNodes[0].InnerText.ToString();
                Console.Write("the auth is " + auth);
                service = xnRoot.ChildNodes[1].InnerText.ToString();
                Console.Write("the service is " + service);
                photo = xnRoot.ChildNodes[3].InnerText.ToString();
                Console.Write("the photo is " + photo);
                photot = xnRoot.ChildNodes[4].InnerText.ToString();
                Console.Write("the photot is " + photot);
                verify_rlt = xnRoot.ChildNodes[2].InnerText.ToString();
                Console.Write("the verify_rlt is " + verify_rlt);
                int photoCount = xd.GetElementsByTagName("Photos").Count;
                for (int i = 0; i < photoCount; i++) {
                                    Console.Write("the photo" + i + " is " +
xnRoot.ChildNodes[5 + i].InnerText.ToString());
                                        string       encode_format       =
xnRoot.ChildNodes[5 + i].InnerText.ToString();
                    //将 base64 字符串解码并保存图片文件到本地
                      byte[]                    bytes                    =
Convert.FromBase64String(encode_format);
```

```
                                    FileStream fs = new
FileStream(@"d:\testPic.jpg", FileMode.Append);
                    fs.Write(bytes, 0, bytes.Length);
                                        fs.Close();
                }
            }
        }catch(Exception ex){
            Console.Write(ex.StackTrace);
        }
    }
}
```

3、PHP 版

```
<?php
/**
 * @desc 综合业务平台--查询 API
 * @author harvey
 * @since 2010-11-20
 *
 */
include_once './SynPlat/DES.php';
class SynPlatAPI {

    /**
     * 取得数据
     * @param string $type   查询类型
     * @param string $param  查询参数
     * @return string
     */
    function getData($type, $param) {
        include './SynPlat/config.php';

        $DES = new DES ( $Key, $iv );
```

```
        try {
            $soap = new SoapClient ( $wsdlURL );
        } catch ( Exception $e ) {
            return "Linkerror";
        }

        //var_dump ( $soap->__getTypes () );
        //@todo 加密数据
        $partner = $DES->encrypt ( $partner );
        $partnerPW = $DES->encrypt ( $partnerPW );
        $type = $DES->encrypt ( $type );
        //先将中文转码
        $param = mb_convert_encoding ( $param, "GBK", "UTF-8" );
        $param = $DES->encrypt ( $param );
        $params = array ("userName_" => $partner, "password_" => $partnerPW,
"type_" => $type, "param_" => $param );
        //请求查询
        $data = $soap->querySingle ( $params );

        //@todo 解密数据
        $resultXML = $DES->decrypt ( $data->querySingleReturn );
        $resultXML = mb_convert_encoding ( $resultXML, "UTF-8", "GBK" );
        return $resultXML;
    }

    /**
     * 格式化参数
     * @param array $params  参数数组
     * @return string
     */
    function formatParam($queryType, $params) {
        include './SynPlat/config.php';
        if (empty ( $supportClass [$queryType] )) {
```

```
            return - 1;
        }
        $keys = array ();
        $values = array ();
        foreach ( $params as $key => $value ) {
            $keys [] = $key;
            $values [] = strtoupper ( $value );
        }
        $param = str_replace ( $keys, $values, $supportClass [$queryType] );
        return $param;
    }

    /**
     * 取得生日（由身份证号）
     * @param int $id 身份证号
     * @return string
     */
    function getBirthDay($id) {
        switch (strlen ( $id )) {
            case 15 :
                $year = "19" . substr ( $id, 6, 2 );
                $month = substr ( $id, 8, 2 );
                $day = substr ( $id, 10, 2 );
            break;
            case 18 :
                $year = substr ( $id, 6, 4 );
                $month = substr ( $id, 10, 2 );
                $day = substr ( $id, 12, 2 );
            break;
        }
        $birthday = array ('year' => $year, 'month' => $month, 'day' => $day );
        return $birthday;
    }
```

```
    /**
     * 取得性别（由身份证号）--可能不准
     * @param int $id 身份证号
     * @return string
     */
    function getSex($id) {
        switch (strlen ( $id )) {
            case 15 :
                $sexCode = substr ( $id, 14, 1 );
            break;
            case 18 :
                $sexCode = substr ( $id, 16, 1 );
            break;
        }
        if ($sexCode % 2) {
            return "男";
        } else {
            return "女";
        }
    }

    /**
     * 格式化数据
     * @param string $type
     * @param srring $data
     * @return array
     */
    function formatData($type, $data) {
        switch ($type) {
            case "1A020201" :
                $detailInfo = $data ['policeCheckInfos'] ['policeCheckInfo'];
                $birthDay = $this->getBirthDay ( $detailInfo
['identitycard'] );
```

```
                $sex = $this->getSex ( $detailInfo ['identitycard'] );
                $info = array (
                        'name' => $detailInfo ['name'],
                        'identitycard' => $detailInfo ['identitycard'],
                        'sex' => $sex,
                        'compStatus' => $detailInfo ['compStatus'],
                        'compResult' => $detailInfo ['compResult'],
                        'policeadd' => $detailInfo ['policeadd'],
                        //'checkPhoto' => $detailInfo ['checkPhoto'],
                        'birthDay' => $birthDay,
                        'idcOriCt2' => $detailInfo ['idcOriCt2'],
                        'resultStatus' => $detailInfo ['compStatus'] );
            break;
            default :
                $info = array (false );
            break;
        }
        return $info;
    }
}
```

DES.php

```
<?php
class DES {
    var $key;
    var $iv; //偏移量

    function DES($key, $iv = 0) {
```

```
        $this->key = $key;
        if ($iv == 0) {
            $this->iv = $key;
        } else {
            $this->iv = $iv;
        }
    }

    //加密
    function encrypt($str) {
        $size = mcrypt_get_block_size ( MCRYPT_DES, MCRYPT_MODE_CBC );
        $str = $this->pkcs5Pad ( $str, $size );

        $data = mcrypt_cbc ( MCRYPT_DES, $this->key, $str, MCRYPT_ENCRYPT,
$this->iv );
        //$data=strtoupper(bin2hex($data)); //返回大写十六进制字符串
        return base64_encode ( $data );
    }

    //解密
    function decrypt($str) {
        $str = base64_decode ( $str );
        //$strBin = $this->hex2bin( strtolower($str));
        $str = mcrypt_cbc ( MCRYPT_DES, $this->key, $str, MCRYPT_DECRYPT,
$this->iv );
        $str = $this->pkcs5Unpad ( $str );
        return $str;
    }

    function hex2bin($hexData) {
        $binData = "";
        for($i = 0; $i < strlen ( $hexData ); $i += 2) {
            $binData .= chr ( hexdec ( substr ( $hexData, $i, 2 ) ) );
        }
```

```
        return $binData;
    }
    function pkcs5Pad($text, $blocksize) {
        $pad = $blocksize - (strlen ( $text ) % $blocksize);
        return $text . str_repeat ( chr ( $pad ), $pad );
    }
    function pkcs5Unpad($text) {
        $pad = ord ( $text {strlen ( $text ) - 1} );
        if ($pad > strlen ( $text ))
            return false;
        if (strspn ( $text, chr ( $pad ), strlen ( $text ) - $pad ) != $pad)
            return false;
        return substr ( $text, 0, - 1 * $pad );
    }
}
?>
```

Config.php

```
<?php
/*
 * 综合业务平台配置文件
 * @author harvey
 * @since 2010-11-20
 */
$wsdlURL = "http://gboss.id5.cn/services/QueryValidatorServices?wsdl";
$partner = "***";
$partnerPW = "***";
$Key = "12345678";
$iv = "12345678";
$supportClass = array ("1A020201" => "Name,CardNum" );
$batchPad = "";
?>
```