|  |
| --- |
| 同程网络科技股份有限公司 |
| **同程旅游API传输控制协议** |
| [Ver1.0.4, 20111128102912] |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本号** | **版本号(日期)** | **修订记录** | **修订人** | **修订日期** |
| v1.0.0 | 20110225160930 | 文档创建 | 李久健 | 2011-2-25 |
| v1.0.1 | 20110228174015 | 定义错误类型与错误代码，确定平台编码方案，确定请求与响应格式 | 李久健 | 2011-2-28 |
| V1.0.2 | 20110303090115 | 重定义错误代码 | 李久健 | 2011-3-3 |
| V1.0.3 | 20110917101530 | 完善文档结构 | 李久健 | 2011-9-17 |
| v1.0.4 | 20111128102912 | 1. 简化请求报文头部参数:去掉[actionCode][transactionID][srcSysId][dscSysId] 2. 更新数字签名算法 | 王纪虎 | 2011-11-29 |

目录

[1 协议框架 1](#_Toc310350656)

[2 协议层次结构 1](#_Toc310350657)

[2.1 协议数据格式 1](#_Toc310350658)

[2.2 会话控制格式 2](#_Toc310350659)

[2.2.1 请求 3](#_Toc310350660)

[2.2.2 应答 4](#_Toc310350661)

[3 操作结果编码 4](#_Toc310350662)

[附录 6](#_Toc310350663)

[A. 协议数据格式示例 6](#_Toc310350664)

[a) 请求信息示例 6](#_Toc310350665)

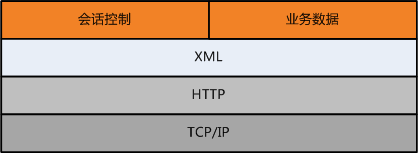
[b) 应答信息示例 6](#_Toc310350666)

[B. 同程旅游API 数字签名算法 7](#_Toc310350667)

# 协议框架

同程旅游API传输控制协议(TCAPITCP)定义实现渠道接触系统或业务平台与同程旅游API(TCAPI)进行交互的数据格式，TCAPITCP协议数据格式采用XML标准规范，采用XML数据格式以确保不同异构系统之间的交互访问。同程旅游API传输控制协议采用松耦合方式，传输协议支持HTTP。认证接口调用采用同步方式。

TCAPITCP协议的结构如下图所示：



**[结构]**

TCAPITCP协议包括会话控制及业务数据两个部分，其中会话控制包含系统交互时所需要的系统级的信息，业务数据包含渠道接触系统或业务平台与同程旅游API交互时所需的信息；在技术上基于标准的HTTP调用实现数据交换，采用同步调用方式。

**[数据表达方式]**

TCAPITCP协议的数据表示方式都采用标准的XML格式。

**[标准技术版本]**

* TCAPITCP协议当前版本所采用的标准HTTP协议版本为1.1。
* TCAPITCP协议当前版本所采用的标准XML版本为1.0。

# 协议层次结构

## 协议数据格式

TCAPITCP协议的数据结构都采用XML方式表示，包括XML Schema文档和XML数据的描述文档(用于协议的理解和使用)。对于每个XML元素的描述包括：

* 父元素名称：表示父节点标识
* 元素名称：表示当前节点标识
* 约束：是否必填，出现次数
* 类型：描述元素的数据对象类型
* 长度：数据的长度，用于物理存储的表示
* 描述
* 取值说明

其中约束、类型、长度的特殊符号说明如下表：

|  |  |  |
| --- | --- | --- |
| **格式类型** | **格式符号** | **说明** |
| 约束 | ? | **0..1** |
| \* | **0..n** |
| + | **1..n** |
| 1 | **1** |
| 类型 | XmlNode | 父节点，并无数据类型 |
| XmlAttr, Type |  |
| string | 字符串，采用String方式表示 |
| DateTime | 日期，必要时需添加格式说明 |
| Decimal |  |
| int |  |
| long |  |
| 长度 | Fn | 固定长度，n个字节 |
| Vn | 非固定长度，最长为n个字节 |

XML数据的描述文档格式约定：

* 在请求/应答信息的描述文档中，省略对根节点(request或response)、报文头节点(header)、报文体节点(body)的描述；
* XML文档非叶子节点的描述，采用淡绿色背景突出显示，如 　　 ；XML文档的节点属性的描述，采用淡黄色背景突出显示，如 　　 ；对XML文档的叶子节点的描述不设背景。

## 会话控制格式

TCAPITCP协议包含会话控制及业务数据两部分，具体XML结构格式如下：

**请求**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **父元素名称** | **元素名称** | **约束** | **描述** | **取值说明** |
| -- | request | 1 |  | 请求request |
| request | header | 1 | 会话控制 | 用于存储服务请求的公共部分和消息的控制数据 |
| request | body | ? | 业务数据 | 服务请求的业务数据，即请求内容。 |

**应答**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **父元素名称** | **元素名称** | **约束** | **描述** | **取值说明** |
| -- | request | 1 |  | 应答response |
| response | header | 1 | 会话控制 | 用于存储服务应答的公共部分和消息的控制数据 |
| response | body | ? | 业务数据 | 服务应答的业务数据，即应答内容。 |

其中会话控制信息是用于控制协议数据交换会话所需要的系统级的信息。会话控制格式如下表所示：



会话控制包括请求/应答信息、安全签名信息等，其中：

* + 请求/应答信息：包括请求/应答类型、请求/应答代码、请求/应答描述；
  + 安全签名：标识不同系统／平台身份的不可抵赖的依据；

会话控制相关元素编码规则如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **名称** | **编码规则** | **说明** |
| 1 | 协议版本号 | 版本标签日期 | 日期格式为yyyyMMddHHmmss |
| 2 | 安全签名 | 系统验证的数字签名 | 详细请看附录B |

### 请求

| **序号** | **父元素名称** | **元素名称** | **约束** | **类型** | **长度** | **描述** | **取值说明** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | request | header | 1 | -- |  | [请求]控制协议头部 |  |
| 1.1 | header | version | 1 | String | F14 | 协议当前使用的版本号。 | 发起方请求时填写；同一交易的应答与请求版本号始终一致 |
| 1.2 | header | accountID | 1 | String | V64 | 帐户标识号 | 同程网为接口调用方所分配的帐户标识号 |
| 1.3 | header | serviceName | 1 | String | V128 | 服务名称 | 请求服务名称，对应接口中的处理方法。 |
| 1.4 | header | digitalSign | ? | String | V128 | 发起方(系统/平台)签名 | 数字签名，发起方(系统/平台)签名  用于系统间的安全认证。根据发起方ID和一组关键参数进行签名。 |
| 1.5 | header | reqTime | 1 | String | F23 | 请求时间，时间格式字符串：yyyy-MM-dd HH:mm:ss.fff | 发起方(系统/平台)用请求方的主机时间填写请求时间(reqTime) |
| 2 | request | body | 1 | -- |  | [请求]控制协议数据部分 |  |
| 2.1 | body | subElement | ? | -- |  | 请求所需传输的数据 | 请求所需传输的数据，以XML形式组织 |

### 应答

| **序号** | **父元素名称** | **元素名称** | **约束** | **类型** | **长度** | **描述** | | **取值说明** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | response | header | 1 | -- |  | [应答]控制协议头部 | |  |
| 1.1 | header | actionCode | 1 | String | F1 | 应答标识 | | 为应答会话控制部分，值为1； |
| 1.2 | header | rspType | 1 | String | F1 | 应答/错误类型 | | 应答或错误类型，取值详见主数据中的编码 |
| 1.3 | header | rspCode | ? | String | F4 | 应答/错误代码 | | 应答或错误代码，取值详见主数据中的编码 |
| 1.4 | header | rspDesc | ? | String | V128 | 应答/错误描述 | | 应答或错误描述 |
| 1.5 | header | transactionID | 1 | String | F23 | 交易流水号 | | 落地方应答时原值回填(发起方的请求流水号)。 |
| 1.6 | header | rspTime | 1 | String | F14 | 应答时间，时间格式字符串：yyyy-MM-dd HH:mm:ss.fff | | 落地方(系统 / 平台)用应答方的主机时间填写应答时间(rspTime)； |
| 2 | response | body | 1 | -- |  | | [应答]控制协议数据部分 |  |
| 2.1 | body | subElement | ? | -- |  | | 响应所需传输的数据 | 响应所需传输的数据，以XML形式组织 |

# 操作结果编码

|  |  |  |
| --- | --- | --- |
| 错误类型及编码 | 应答代码(四位) | 描述 |
| 成功：0 | 0000 | 查询成功，有结果 |
| 0001 | 查询成功，无结果 |
| 参数错误：1 | 1000 | 参数错误，未明确具体类别 |
| 1001 | 缺少参数 |
| 1002 | 参数类型错误 |
| 1003 | 参数值错误，如超出取值范围等 |
| …… |  |
| 数据错误：2 | 2000 | 数据错误，未明确具体类别 |
| 2001 | 连接数据库失败 |
| 2002 | 添加失败 |
| 2003 | 查询失败 |
| 2004 | 更新失败 |
| 2005 | 删除失败 |
| 2006 | 查询超出限定次数 |
| …… |  |
| 网络错误：3 | 3000 | 网络错误，未明确具体类别 |
| 3001 | 网络连接失败 |
| 3002 | 拒绝访问(未授权) |
| 3003 | 网络连接超时 |
| …… |  |
| 扩展标识：4~6 | 待扩展 |  |
|  |  |
|  |  |
| 执行异常：7 | 7000 | 程序执行时产生未处理的异常，程序出错 |
| …… |  |
| 认证错误：8 | 8000 | 认证错误，未明确具体类别 |
| 8001 | 帐号不存在或者密码错误 |
| 8002 | 认证平台尚未注册或对外提供服务 |
| 8003 | 业务平台尚未接入认证平台 |
| 8004 | 断言查询错误[在特定机制下的查询] |
| …… |  |
| 报文错误：9 | 9000 | 报文错误，未明确具体类别 |
| 9001 | 流水号重复 |
| 9002 | 请求消息报文控制报头格式错误 |
| 9003 | 请求消息报文业务报体格式错误 |
| 9004 | 应答消息报文控制报头格式错误 |
| 9005 | 应答消息报文业务报体格式错误 |
| 9006 | 请求报文格式错误 |
| 9007 | 应答报文格式错误 |
| 9008 | 签名验证错误 |
| 9009 | 未定义服务名称 |
| 9010 | 服务名称错误或尚未提供指定服务 |
| …… |  |

# 附录

## 协议数据格式示例

### 请求信息示例

<?xml version="1.0" encoding="utf-8"?>

<request>

  <header>

    <version>20111128102912</version>

    <accountID>{guid}</accountID>

    <serviceName>GetHotelList</serviceName>

    <digitalSign>{digitalSign\_\_\_\_\_\_\_1}</digitalSign>

    <reqTime>{reqTime}</reqTime>

  </header>

  <body>

    <cityId>{cityId}</cityId>

  </body>

</request>

### 应答信息示例

<?xml version="1.0" encoding="utf-8"?>

<response>

  <header>

    <actionCode>1</actionCode>

    <rspType>0</rspType>

    <rspCode>0000</rspCode>

    <rspDesc><![CDATA[查询成功]]></rspDesc>

    <transactionID>00004201109171251020123</transactionID>

    <digitalSign>{digitalSign\_\_\_\_\_\_1}</digitalSign>

    <rspTime>{rspTime}</rspTime>

  </header>

  <body>

    <hotelList page="1" pageSize="10" totalPage="235" totalCount="2342">

      <hotel Id="123001">

        <name>香港金涛酒店</name>

        <lowestPrice>380.00</lowestPrice>

        <address><![CDATA[佐敦道18~24号鸿运大厦3楼C&D室]]></address>

      </hotel>

      <hotel Id="123002">

        <name>赣州热泉河酒店</name>

        <lowestPrice>420.00</lowestPrice>

        <address><![CDATA[安远县三百山<东江源温泉旅游度假区>]]></address>

      </hotel>

      <!--若干酒店数据节点-->

      <!--<hotel>...</hotel>-->

    </hotelList>

  </body>

</response>

## 同程旅游API 数字签名算法

C# 版本

/// <summary>

/// 加密专用类

/// </summary>

public class Encryption

{

    /// <summary>

    /// 获取字符数组的MD5加密值

    /// </summary>

    /// <param name="sortedArray">待计算MD5哈希值的输入字符数组</param>

    /// <param name="key">密钥</param>

    /// <param name="charset"></param>

    /// <returns>输入字符数组的MD5哈希值</returns>

    public static string GetMD5ByArray(string[] sortedArray, string key, string charset)

    {

        StringBuilder builder = new StringBuilder();

        for (int i = 0; i < sortedArray.Length; i++)

        {

            if (i == sortedArray.Length - 1)

            {

                builder.Append(sortedArray[i]);

            }

            else

            {

                builder.Append(sortedArray[i] + "&");

            }

        }

        builder.Append(key);

        return GetMD5(builder.ToString(), charset);

    }

    /// <summary>

    /// MD5加密

    /// </summary>

    /// <param name="input">待计算MD5哈希值的输入字符串</param>

    /// <param name="charset">输入字符串的字符集</param>

    /// <returns>输入字符串的MD5哈希值</returns>

    public static string GetMD5(string input, string charset)

    {

        MD5 md5 = new MD5CryptoServiceProvider();

        byte[] data = md5.ComputeHash(Encoding.GetEncoding(charset).GetBytes(input));

        StringBuilder builder = new StringBuilder(32);

        for (int i = 0; i < data.Length; i++)

        {

            builder.Append(data[i].ToString("x2"));

        }

        return builder.ToString();

    }

}

/// <summary>

/// 通用类

/// </summary>

public static class Common

{

    /// <summary>

    /// 数组排序（冒泡排序法）

    /// </summary>

    /// <param name="originalArray">待排序字符串数组</param>

    /// <returns>经过冒泡排序过的字符串数组</returns>

    public static string[] BubbleSort(string[] originalArray)

    {

        int i, j;

        string temp;

        bool exchange;          //交换标志

        for (i = 0; i < originalArray.Length; i++)  //最多做R.Length-1趟排序

        {

            exchange = false;   //本趟排序开始前，交换标志应为假

            for (j = originalArray.Length - 2; j >= i; j--)

            {

                //判断交换条件

                if (String.CompareOrdinal(originalArray[j + 1], originalArray[j]) < 0)

                {

                    temp = originalArray[j + 1];

                    originalArray[j + 1] = originalArray[j];

                    originalArray[j] = temp;

                    exchange = true; //发生了交换，故将交换标志置为真

                }

            }

            if (!exchange) //本趟排序未发生交换，提前终止算法

            {

                break;

            }

        }

        return originalArray;

    }

}

public abstract class Validator

{

///<summary>

/// 签名算法[节选]

///</summary>

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

///<returns>true, 通过签名校验； false，未通过签名校验</returns>

private static bool ValidateDigitalSign(Hashtable htValidationParams)

{

if (!htValidationParams.Contains("AccountKey")

   || String.IsNullOrWhiteSpace(htValidationParams["AccountKey"].ToString()))

{

   throw new ApplicationException("缺少API帐户密钥");

}

//API帐户密钥

string accountKey = htValidationParams["AccountKey"].ToString().Trim();

bool flag = false;

List<string>stringList = new List<string>();

if (htValidationParams.Contains("Version"))

stringList.Add(string.Format("Version={0}", htValidationParams["Version"].ToString()));

if (htValidationParams.Contains("AccountID"))

stringList.Add(string.Format("AccountID={0}", Guid.Parse(htValidationParams["AccountID"].ToString())));

if (htValidationParams.Contains("ServiceName"))

stringList.Add(string.Format("ServiceName={0}", htValidationParams["ServiceName"].ToString()));

if (htValidationParams.Contains("ReqTime")

&& htValidationParams["ReqTime"] != null)

stringList.Add(string.Format("ReqTime={0}", ((DateTime)htValidationParams["ReqTime"]).ToString("yyyy-MM-dd HH:mm:ss.fff")));

string[] originalArray = stringList.ToArray();

string[] sortedArray = Common.BubbleSort(originalArray);

string otherDigitalSign = Encryption.GetMD5ByArray(sortedArray, accountKey, "utf-8");

if (otherDigitalSign.Equals(htValidationParams["DigitalSign"].ToString(), StringComparison.InvariantCultureIgnoreCase))

{

flag = true;

}

else

{

flag = false;

}

return flag;

}

}

Java 版

/// <summary>

/// 获取数组加密值

/// </summary>

/// <param name="sortedArray"></param>

/// <returns></returns>

**public** **static** String GetMD5ByArray(String[] sortedArray, String key, String inputCharset)

{

//构造待md5摘要字符串

StringBuilder builder = **new** StringBuilder();

**for** (**int** i = 0; i < sortedArray.length; i++)

{

**if** (i == sortedArray.length - 1)

{

builder.append(sortedArray[i]);

}

**else**

{

builder.append(sortedArray[i] + "&");

}

}

builder.append(key);//此处key为上面的innerKey

**return** *GetMD5*(builder.toString(), inputCharset);

}

/// <summary>

/// md5加密

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

**public** **static** String GetMD5(String s)

{

**return** *GetMD5*(s, "utf-8");

}

/// <summary>

/// MD5加密（已重载）

/// </summary>

/// <param name="input"></param>

/// <param name="inputCharset"></param>

/// <returns></returns>

**public** **static** String GetMD5(String input, String inputCharset) {

String s = **null**;

**char** hexDigits[] = { // 用来将字节转换成 16 进制表示的字符

'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'};

**try**

{

java.security.MessageDigest md = java.security.MessageDigest.*getInstance*( "MD5" );

md.update( input.getBytes(inputCharset) );

**byte** tmp[] = md.digest(); // MD5 的计算结果是一个 128 位的长整数，

// 用字节表示就是 16 个字节

**char** str[] = **new** **char**[16 \* 2]; // 每个字节用 16 进制表示的话，使用两个字符，

// 所以表示成 16 进制需要 32 个字符

**int** k = 0; // 表示转换结果中对应的字符位置

**for** (**int** i = 0; i < 16; i++) { // 从第一个字节开始，对 MD5 的每一个字节

// 转换成 16 进制字符的转换

**byte** byte0 = tmp[i]; // 取第 i 个字节

str[k++] = hexDigits[byte0 >>> 4 & 0xf]; // 取字节中高 4 位的数字转换,

// >>> 为逻辑右移，将符号位一起右移

str[k++] = hexDigits[byte0 & 0xf]; // 取字节中低 4 位的数字转换

}

s = **new** String(str); // 换后的结果转换为字符串

}**catch**( Exception e )

{

e.printStackTrace();

}

**return** s;

}

/// <summary>

/// 数组排序（冒泡排序法）

/// </summary>

/// <param name="originalArray"></param>

/// <returns></returns>

**public** **static** String[] BubbleSort(String[] originalArray)

{

**int** i, j; //交换标志

String temp;

Boolean exchange;

**for** (i = 0; i < originalArray.length; i++) //最多做R.Length-1趟排序

{

exchange = **false**; //本趟排序开始前，交换标志应为假

**for** (j = originalArray.length - 2; j >= i; j--)

{

**if** ( originalArray[j + 1].compareTo(originalArray[j]) < 0)//交换条件

{

temp = originalArray[j + 1];

originalArray[j + 1] = originalArray[j];

originalArray[j] = temp;

exchange = **true**; //发生了交换，故将交换标志置为真

}

}

**if** (!exchange) //本趟排序未发生交换，提前终止算法

{

**break**;

}

}

**return** originalArray;

}

DigitalSign\_CodeSnippet

String[] originalArray={"Version=" + version,"AccountID=" + accountId, "ServiceName="+methodName, "ReqTime="+ reqTime };

String[] sortedArray = Tools.*BubbleSort*(originalArray);

String checkvalue = Tools.*GetMD5ByArray*(sortedArray, *ACCOUNTKEY*, "utf-8");