CRM登录验证接口文档

接口概述

`AICheckLogin` 接口用于验证用户登录信息，通过验证后返回登录状态和相关信息(预留)。

需求XQ_250818014：CRM登陆认证api--用于朗珈知识库（上海朗珈软件有限公司）

接口基本信息

- **接口地址**: `/api/User/AICheckLogin`
- **请求方式**: POST
- **请求格式**: JSON
- **响应格式**: JSON

请求参数

请求体参数

参数名	类型	必填	描述
Account	string	是	用户账号
Token	string	是	加密令牌，格式请参考Token生成规则

加密令牌，格式请参考Token生成规则

Token生成规则

1. 组成格式: `Account|Password|Timestamp`
   - Account: 用户账号
   - Password: 用户密码
   - Timestamp: Unix时间戳（秒级）
   - 规则:拼接字符串后,使用AES加密,加密方法以及时间戳方法已提供;
2. Timestamp参考:

```
DateTime dateTime = DateTime.UtcNow;
DateTimeOffset dateTimeOffset = new DateTimeOffset(dateTime);
long timestamp = dateTimeOffset.ToUnixTimeSeconds();
```

3. 加密方式参考:

代码块

```
public class AESHelper
{
    /// <summary>
    /// 密钥
    /// </summary>
    public static string Key { get; set; } = "l1o2g3e4nE1234@!";
    /// <summary>
    /// 偏移量（IV）
    /// </summary>
    public static string IV { get; set; } = "4s3c2a1p$llogene";

    /// <summary>
    /// AES加密
    /// </summary>
    /// <param name="text">明文字符串</param>
    /// <param name="key">密钥</param>
    /// <returns>密文</returns>
    public static string AESEncrypt(string text, string key = null)
    {
        string strKey = string.IsNullOrWhiteSpace(key) ? Key : key;
        RijndaelManaged rijndaelCipher = new RijndaelManaged();
        rijndaelCipher.Mode = CipherMode.CBC;
        rijndaelCipher.Padding = PaddingMode.PKCS7;
        rijndaelCipher.KeySize = 128;
        rijndaelCipher.BlockSize = 128;
        byte[] pwdBytes = System.Text.Encoding.UTF8.GetBytes(strKey);
        byte[] keyBytes = new byte[16];
        int len = pwdBytes.Length;
        if (len > keyBytes.Length) len = keyBytes.Length;
        System.Array.Copy(pwdBytes, keyBytes, len);
        rijndaelCipher.Key = keyBytes;
        byte[] ivBytes = System.Text.Encoding.UTF8.GetBytes(IV);
        rijndaelCipher.IV = ivBytes;
        ICryptoTransform transform = rijndaelCipher.CreateEncryptor();
        byte[] plainText = Encoding.UTF8.GetBytes(text);
        byte[] cipherBytes = transform.TransformFinalBlock(plainText, 0,
plainText.Length);
        return Convert.ToBase64String(cipherBytes);
```

```
38      }
39
40      /// <summary>
41      /// AES解密
42      /// </summary>
43      /// <param name="text">加密字符串</param>
44      /// <param name="key">密钥</param>
45      /// <returns>明文</returns>
46      public static string AESDecrypt(string text, string key = null)
47      {
48          string strKey = string.IsNullOrWhiteSpace(key) ? Key : key;
49          RijndaelManaged rijndaelCipher = new RijndaelManaged();
50          rijndaelCipher.Mode = CipherMode.CBC;
51          rijndaelCipher.Padding = PaddingMode.PKCS7;
52          rijndaelCipher.KeySize = 128;
53          rijndaelCipher.BlockSize = 128;
54          byte[] encryptedData = Convert.FromBase64String(text);
55          byte[] pwdBytes = System.Text.Encoding.UTF8.GetBytes(strKey);
56          byte[] keyBytes = new byte[16];
57          int len = pwdBytes.Length;
58          if (len > keyBytes.Length) len = keyBytes.Length;
59          System.Array.Copy(pwdBytes, keyBytes, len);
60          rijndaelCipher.Key = keyBytes;
61          byte[] ivBytes = System.Text.Encoding.UTF8.GetBytes(IV);
62          rijndaelCipher.IV = ivBytes;
63          ICryptoTransform transform = rijndaelCipher.CreateDecryptor();
64          byte[] plainText = transform.TransformFinalBlock(encryptedData, 0,
    encryptedData.Length);
65          return Encoding.UTF8.GetString(plainText);
66      }
67
68      /// <summary>
69      /// MD5加密
70      /// </summary>
71      /// <param name="str">加密字符</param>
72      /// <param name="code">加密位数16/32</param>
73      /// <returns></returns>
74      public static string MD5Hash(string str, int code)
75      {
76          string strEncrypt = string.Empty;
77          if (code == 16)
78          {
79              MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();
80              strEncrypt =
    BitConverter.ToString(md5.ComputeHash(Encoding.Default.GetBytes(str)), 4,
    8).Replace("-", "");
81          }
```

```
82
83          if (code == 32)
84          {
85              MD5 md5 = MD5.Create();//实例化一个md5对像
86                                      // 加密后是一个字节类型的数组，这里要注意编码
    UTF8/Unicode等的选择
87              byte[] s = md5.ComputeHash(Encoding.UTF8.GetBytes(str));
88              // 通过使用循环，将字节类型的数组转换为字符串，此字符串是常规字符格式化所得
89              for (int i = 0; i < s.Length; i++)
90              {
91                  // 将得到的字符串使用十六进制类型格式。格式后的字符是小写的字母，如果使
    用大写（X）则格式后的字符是大写字符
92
93                  strEncrypt = strEncrypt + s[i].ToString("X2");
94
95              }
96          }
97
98          return strEncrypt.ToLower();
99      }
100 }
```

响应参数

响应体结构

代码块

```
1  {
2      "Message": "string",
3      "Success": "boolean",
4      "Code": "string",
5      "Content": "object"
6  }
```

响应码说明

响应码	说明
1	登录验证成功
-1	账号或Token为空
-2	Token解密失败
-3	Token格式不正确
-4	Token中的账号与请求账号不匹配
-5	Token已过期（有效期10分钟）
-6	用户不存在
-7	用户登录失败次数超过5次
-8	密码不正确
-99	系统异常

响应码

说明

1

登录验证成功

-1

账号或Token为空

-2

Token解密失败

-3

Token格式不正确

-4

Token中的账号与请求账号不匹配

-5

Token已过期（有效期10分钟）

-6

用户不存在

-7

用户登录失败次数超过5次

-8

密码不正确

-99

系统异常

请求示例

代码块

```
{
    "Account": "testuser",
    "Token": "o0lpO07BiCRPrxeyEitc97b/BrJjvyWryvKf/56RbXI="
}
```

响应示例

成功响应

代码块

```
{
    "Message": "登录验证成功! ",
    "Success": true,
    "Code": "1",
    "Content": null
}
```

失败响应

代码块

```
{
    "Message": "登录验证失败! ",
    "Success": false,
    "Code": "-5",
    "Content": null
}
```

注意事项

1. Token有效期: 10分钟（从Timestamp时间开始计算）
2. 时间戳: 使用Unix时间戳（秒级）