ارسال پیام دریافتی ازکاربر به اکسترنال CP

نسخه ۲,۱,۰,۰

تاریخ ۱۳۹۶/۱۱/۲۳

برای دریافت پیام کاربر باید یک post Api بر روی پروتکل http وجود داشته باشد. این آدرس Api باید در اختیار شرکت فناپ قرار بگیرد.

پس ازدریافت پیام کاربر , سامانه پیام رسان فناپ ,اطلاعات زیر را در اختیار شرکت مشتری قرار می دهد.

Http request body:

بدنه پیام دارای object به فرمت زیر است:

Muid	شناسه منحصر به فردی که توسط پیام رسان فناپ به پیام اختصاص داده شده است.
ReceiveTime	زمان دریافت پیام توسط سامانه (Iso 8601) به فرمت : universal time "yyyy-MM-ddTHH:mm:ss.fffZ"
AccountId	شناسه کاربر
ChannelType	درگاهی که پیام کاربر از آن دریافت شده است، شامل مقادیر: Pardis, Imi, Mtn, Rightel است.
Channel	سرشماره ای که پیام کاربر از آن دریافت شده است
MessageType	نوع پیام دریافتی که دارای مقادیر: Content, Subscription, Unsubscription, OnDemand است.
Message	متن پیام ارسال شده توسط کاربر
SID	شناسه منحصر به فرد مشتری که توسط شرکت فناپ در اختیار مشتری قرار گرفته و به منظور شناسایی مشتری از آن استفاده میکند.
Signature	پیام امضا شده توسط شرکت فناپ

در صورتی که متقاضی دریافت شماره کاربر هستندinput، مقادیر زیر نیز ارسال می گردد:

شماره تلفن کاربر	UserNumber

پارامتر نوع پیام دریافتی دارای یکی از مقادیر زیر است :

- Content عادی
- Subscription عضویت
- unSubscription لغو عضویت
- OnDemand آندیمند

پارامتر Signature:

برای احراز هویت و اثبات صحت پیام می‌باشد. فناپ با توجه به پارامترهای بدنه‌ی پیام ، پیام فرمت شده‌ای را می‌سازد که همان پیام را توسط کلید خصوصی خود امضا می‌کند و در پارامتر Signature قرار می‌دهد . شرکت خصوصی محتوای این پیام را با کلید عمومی فناپ (کلید عمومی فناپ را همراه داکیومنت دریافت کنید)، بازگشایی می‌کند و چک می‌کند که پیام ارسالی از جانب فناپ باشد.

پیام فرمت شده بدین ترتیب ایجاد شده است (توضیح آنکه، مقادیر داخل براکت با توجه به مقادیر پارامترهای نظیر در بدنه ی پیام به صورت جدا شده با ویرگول، پر می شوند و ترتیب از چپ به راست حتما رعایت شود):

```
[ReceiveTime],[SID],[ChannelType],[Channel],[Muid],[Message],[MessageType],[AccountId]
```

شرکت فناپ منتظر پاسخ فراخوانی مشتری نمی ماند و در صورتی که مشتری قصد ارسال پیام به کاربر را دارد باید وب سرویس ارسال پیام به مشتری پیاده سازی شود و از طریق آن پیام ها ارسال شود. سرویس ارسال پیام به مشتری

نیازمند یک کلید نامتقارن است که برای امضا و احراز هویت مشتری مورد استفاده قرار میگیرد، بنابراین کلید عمومی آن در فرمت XML باید در اختیار شرکت فناپ قرار بگیرد.

**برای اینکه آن شرکت بتواند پیام دریافتی از سامانه‌ی پیام‌رسان فناپ را در اختیار سامانه‌ی پیکو (سامانه‌ی پرداخت فناپ) قرار دهد مقدار پارامتر signature و پیام فرمت شده ای را که از بدنه پیام به منظور احراز هویت ساخته اید به پیکو ارسال کنید.

پیوست ۱:

نحوه sign کردن فناپ :

الگوریتم مورد استفاده برای asymmetric cryptography ، الگوریتم RSA است که الگوریتم hash آن نیز SHA1 است نمونه کد در زیر آمده است.

```
public static string Sign(string key, string text)
    {
        using (var rsaProvider = new RSACryptoServiceProvider(CspParams))
        {
            rsaProvider.FromXmlString(key);

            var plainBytes = Encoding.UTF8.GetBytes(text);

                var encryptedBytes = rsaProvider.SignData(plainBytes, new
                                    SHA1CryptoServiceProvider());

            return Convert.ToBase64String(encryptedBytes);
        }
    }
```

پیوست۲:

نمونه کد احراز هویت :

```
public static bool Check(string key, string signedText, string text)
      {
         if (string.IsNullOrWhiteSpace(text)) return false;
         try
         {
            // Select target CSP
            var cspParams = new CspParameters { ProviderType = 1 };
            // PROV_RSA_FULL
            //cspParams.ProviderName; // CSP name
            var rsaProvider = new RSACryptoServiceProvider(cspParams);


            // Import private/public key pair
            rsaProvider.FromXmlString(key);

            var encryptedBytes = Convert.FromBase64String(signedText);
            var plainInput = Encoding.UTF8.GetBytes(text);

            // Decrypt text
var check =
rsaProvider.VerifyData(plainInput, new SHA1CryptoServiceProvider(), encryptedB
ytes);
            return check;
         }
         catch (Exception exception)
         {
            Log.Error(exception.Message, exception);
            return false;
         }

}
```

```
public static bool Check(string key, string signedText, string text)
      {
         if (string.IsNullOrWhiteSpace(text)) return false;
         try
         {
            // Select target CSP
            var cspParams = new CspParameters { ProviderType = 1 };
            // PROV_RSA_FULL
            //cspParams.ProviderName; // CSP name
            var rsaProvider = new RSACryptoServiceProvider(cspParams);


            // Import private/public key pair
            rsaProvider.FromXmlString(key);

            var encryptedBytes = Convert.FromBase64String(signedText);
            var plainInput = Encoding.UTF8.GetBytes(text);

            // Decrypt text
var check =
rsaProvider.VerifyData(plainInput, new SHA1CryptoServiceProvider(), encryptedB
ytes);
            return check;
         }
         catch (Exception exception)
         {
            Log.Error(exception.Message, exception);
            return false;
         }

}
```

```
public static bool Check(string key, string signedText, string text)
      {
         if (string.IsNullOrWhiteSpace(text)) return false;
         try
         {
            // Select target CSP
            var cspParams = new CspParameters { ProviderType = 1 };
            // PROV_RSA_FULL
            //cspParams.ProviderName; // CSP name
            var rsaProvider = new RSACryptoServiceProvider(cspParams);


            // Import private/public key pair
            rsaProvider.FromXmlString(key);

            var encryptedBytes = Convert.FromBase64String(signedText);
            var plainInput = Encoding.UTF8.GetBytes(text);

            // Decrypt text
var check =
rsaProvider.VerifyData(plainInput, new SHA1CryptoServiceProvider(), encryptedB
ytes);
            return check;
         }
         catch (Exception exception)
         {
            Log.Error(exception.Message, exception);
            return false;
         }

}
```