2013

NetCORE Solutions Pvt. Ltd.

[2FA- API DOC]

Missed call for authentication

Version	Created by	Reviewed by
1.0	Vivek P. & Gaurav Paneri	Sridhar P.

Contents

1. Getting Started

1.1 Introduction

2fa is a web-service which helps Enterprises add two-factor authentication to their applications. This service uses 'missed call' as the second medium for authentication.

1.2 Before you start

Before implementing 'Two Factor Authentication services' for your Enterprise application, you must contact the **netCORE admin** and register your application.

As part of the registration process you will receive the following:

- **Secured2fa key::** This public key is used by netCORE to identify your application.
  Note:: This key is to be sent in each request to netCORE's API.

- **Secret key::** This private key will be known only to your enterprise and netCORE. It is used in the encryption of a signature that is used to secure communication between your enterprise and netCORE.
  Note:: This key is **NOT TO BE SENT** in any request to netCORE's API.

- **Origin code ::** This is the number on which your enterprise applications end-users will send missed calls to. E.g. 022- 3025XXXX

1.3 Target Audience

The target audiences for this document are:-

- **Application development teams** of Enterprises, who will implement this web-service in their applications.
- **Quality Assurance teams** of Enterprises, who will verify the implementation of this web-service in their application.

1.4 2fa API Overview :: Basic Concepts

1. What is Two factor authentication?

Two-factor authentication is an approach to authentication which requires the presentation of two authentication factors:

- A knowledge factor ("something the user knows") :: a password
- A possession factor ("something the user has") :: hardware tokens, mobile phones

2. Why should an Enterprise use Two factor authentication to their product/service?

Even if passwords are stolen or hacked, the attacker cannot log in with these passwords without the second tokens.

3. What is netCORE's proposition?

Adds a second factor to your password-enabled applications by using a missed call as 2nd factor of authentication.

1.5 2fa API Overview :: Authentication

netCORE uses **OAuth Version 1.0a** for securing communication between itself and enterprises. Details about **OAuth Versionv1.0a** can be found here: **http://oauth.net/core/1.0a/**

Note :: All implementations of the '2fa API' must use **OAuth Version 1.0a**.

1.6 Sending Requests

The '2fa API V1.0' supports the **POST** method of sending requests to the 2fa server:

The URI is: **https://2fa.co.in/api/**

1.7 Data Formats

Each request or response body will contain XML data which should be passed as RAW POST DATA.

2. Using the API

2.1 Introduction

While using 2fa, enterprise applications will have to submit a request through the <Create Session API>. The application can get a response by either of two methods i.e. (PULL or PUSH)

- To get a response using PULL method, the application will have to call <Get Session API>
- To get a response using PUSH method, enterprises will have to provide their <Pingback API> to the netCORE admin at the time of registration.
    - <Pingback API> will be called by netCORE on receiving a missed call for a request.

2.2 Identifying your App

netCORE uses the secured2fakey to identify your application. Each request from your application should contain the following:

Identification Parameter:

<secured2fakey>32 bit secured 2fa key</secured2fakey>

2.3 Authentication

netCORE uses a signature to verify if a request is valid or not. Each request from your application should contain the following:

Authentication Parameters:
<signature>A 32-bit unique hash</signature>
<signaturemethod>HMAC-SHA1, HMAC-RSA1 or PLAIN-TEXT</signaturemethod>
<nonce>A unique hash</nonce>
<timestamp>current time stamp of your application</timestamp>

2.4 How to build a 2fa request?

1. The first step is **creating a signature**.

   A signature is a string of 'key=value' pairs separated by '&'. It is encrypted using OAuth Version 1.0a. The keys required to build the string are described below:

   - secured 2fa key
   - secret key
   - signature method

2. Once you have created a signature, you need to **create a request body** in XML with the required parameters specified in the API reference.

2.5 2fa API Reference

2.5.1 Create Session API

Method	URL
POST	https://2fa.co.in/api/createSession

Request

Type	Params	Values
POST	msisdn	number(10digit)
POST	2fasecuredkey	string
POST	origincode	number

POST	timestamp	number
POST	signature	string
POST	nonce	string
POST	signaturemethod	string

Syntax

```
<?xml version='1.0'?>
<request>
        <msisdn>Mobile Number</msisdn>
        <origincode>Origin Code</origincode>
        <secured2fakey>A 32 bit secured 2fa key</secured2fakey>
        <nonce>A unique hash</nonce>
        <signature>A 32 bit unique hash</signature>
        <signaturemethod>Signature Method Used</signaturemethod>
        <timestamp>Current time stamp of your application</timestamp>
</request>
```

Response

Status	Response
200	<status>OK</status> <message>Session has been created</message>
400	<status>NOK</status> <message>Authentication Failed</message>
402	<status>NOK</status> <message> Invalid Request</message>
404	<status>NOK</status> <message> Invalid Secured 2FA key</message>
405	<status>NOK</status> <message>Your Secured 2FA Key Is Inactive</message>
406	<status>NOK</status> <message>Invalid Origin Code</message>
408	<status>NOK</status> <message> Invalid Signature</message>
500	<status>NOK</status> <message> Internal Server Error</message>

Syntax

```
<? xml version='1.0'?>
        <response>
                <statuscode>200</statuscode>
                <status>OK</status>
                <message>Session has been created</message>
```

```
    <sessid>32 bit session id</sessid>
  <secured2fakey>A 32 bit secured 2fa key</secured2fakey>
  <nonce>A unique hash</nonce>
    <signature>A 32 bit unique hash</signature>
    <signaturemethod>Signature Method Used</signaturemethod>
    <timestamp>Current time stamp of your application</timestamp>
</response>
```

2.5.2 Get Session API

Method	URL
POST	https://2fa.co.in/api/getSession

Request

Type	Params	Values
POST	sessid	String
POST	secured2fakey	String
POST	timestamp	number
POST	signature	string
POST	nonce	string
POST	signaturemethod	string
POST	authtoken	string

Syntax

```
<?xml version='1.0'?>
    <request>
        <sessid>32 bit session id</sessid>
        <secured2fakey>A 32 bit secured 2fa key</secured2fakey>
        <nonce>A unique hash</nonce>
          <signature>A 32 bit unique hash</signature>
          <signaturemethod>Signature Method Used</signaturemethod>
          <timestamp>Current time stamp of your application</timestamp>
    </request>
```

Response

Status	Response
200	<message>Mobile number verified successfully</message> <verifystatus>1</verifystatus>

200	<message> Missed call not received </message> <verifystatus>0</verifystatus>
200	<message>Missed call not received and session time out</message> <verifystatus>-1</verifystatus>
200	<message>missed call received but session session timeout </message> <verifystatus>-2</verifystatus>
405	<status>NOK</status> <message>Your Secured 2FA Key Is Inactive</message>
500	<status>NOK</status> <message> Internal Server Error </message>
502	<status>NOK</status> <message> Session Not Found </message>

Syntax

```
<? xml version='1.0'?>
        <response>
                <statuscode>Status Code</statuscode>
                <status>OK</status>
                <message>Mobile number verified successfully</message>
                <verifystatus>Verify Status Code</verifystatus>
                <msisdn>Mobile Number</msisdn>
                <origincode>Origin Code</origincode>
               <secured2fakey>A 32 bit secured 2fa key</secured2fakey>
              <nonce>A unique hash</nonce>
                <signature>A 32 bit unique hash</signature>
                <signaturemethod>Signature Method Used</signaturemethod>
                <timestamp>Current time stamp of your application</timestamp>
        </response>
```

2.5.3 Pingback API

Method	URL
POST	<User API URL>

Response

Status	Response
200	<message>Mobile number verified successfully</message> <verifystatus>Verify Status Code</verifystatus>
200	<message>Missed call received but session time out</message>

	<verifystatus>-2</verifystatus>
200	<message>Invalid transaction because of multiple sessions created </message> <verifystatus>-3</verifystatus>

Syntax

```
<? xml version='1.0'?>
        <response>
                <resp>
                <statuscode>Status Code</statuscode>
                <status>OK</status>
                <message>Mobile number verified successfully</message>
                <verifystatus>Verify Status Code</verifystatus>
                <sessid>32 bit session id</sessid>
                <msisdn>Mobile Number</msisdn>
                <origincode>Origin Code</origincode>
                <secured2fakey>A 32 bit secured 2fa key</secured2fakey>
                <nonce>A unique hash</nonce>
                <signature>A 32 bit unique hash</signature>
                <signaturemethod>Signature Method Used</signaturemethod>
                <timestamp>Current time stamp of your application</timestamp>
                </resp>
        </response>
```

2.6 API Examples

2.6.1 Create Session API

Request

```
<?xml version='1.0'?>
        <request>
                <msisdn>9876543210</msisdn>
                <origincode>9902099020</origincode>
                <secured2fakey>8aa6bc8a7a36f70a90701c9db4d9</secured2fakey>
                <signature>0xde7c9b85b8b78aa6bc8a7a36f70a90701</signature>
                <signaturemethod>HMAC-SHA1</signaturemethod>
                <nonce>b8y7017ht0987</nonce>
                <timestamp>398743265</timestamp>
        </request>
```

Response

```
<? xml version='1.0'?>
	<response>
		<statuscode>200</statuscode>
		<status>OK</status>
		<message>Session has been created</message>
		<sessid>8aa6bc8a7a36f70a90701c9db4d9</sessid>
		<secured2fakey>6543defrdnghtuyterdgfnbtyredytr</secured2fakey>
		<signature>0xde7c9b85b8b78aa6bc8a7a36f70a90701</signature>
		<signaturemethod>HMAC-SHA1</signaturemethod>
		<nonce>b8ytrb86mky46cbrf70a907017ht0987</nonce>
		<timestamp>398743265</timestamp>
	</response>
```

2.6.2 Get Session API

Request

```
<?xml version='1.0'?>
	<request>
		<sessid>8aa6bc8a7a36f70a90701c9db4d9</sessid>
		<secured2fakey>8aa6bc8a7a36f70a90701c9db4d9</secured2fakey>
		<signature>0xde7c9b85b8b78aa6bc8a7a36f70a90701</signature>
		<signaturemethod>HMAC-SHA1</signaturemethod>
		<nonce>b8y7017ht0987</nonce>
		<timestamp>398743265</timestamp>
	</request>
```

Response

```
<? xml version='1.0'?>
	<response>
		<statuscode>200</statuscode>
		<status>OK</status>
		<message>Session has been created</message>
		<sessid>8aa6bc8a7a36f70a90701c9db4d9</sessid>
		<secured2fakey>6543defrdnghtuyterdgfnbtyredytr</secured2fakey>
		<signature>0xde7c9b85b8b78aa6bc8a7a36f70a90701</signature>
		<signaturemethod>HMAC-SHA1</signaturemethod>
		<nonce>b8ytrb86mky46cbrf70a907017ht0987</nonce>
		<timestamp>398743265</timestamp>
	</response>
```

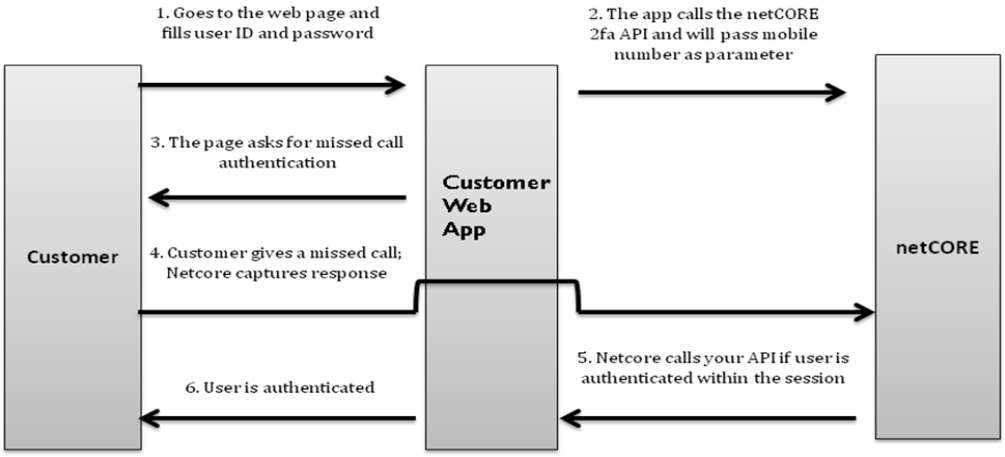
2.6.3 Pingback API

Response

```
<? xml version='1.0'?>
        <response>
                <statuscode>200</statuscode>
                <status>OK</status>
                <message>Mobile number verified successfully</message>
                <verifystatus>1</verifystatus>
                <sessid>8aa6bc8a7a36f70a90701c9db4d9</sessid>
                <secured2fakey>6543defrdnghtuyterdgfnbtyredytr</secured2fakey>
                <signature>0xde7c9b85b8b78aa6bc8a7a36f70a90701</signature>
                <signaturemethod>HMAC-SHA1</signaturemethod>
                <nonce>b8ytrb86mky46cbrf70a907017ht0987</nonce>
                <timestamp>398743265</timestamp>
        </response>
```

3. Glossary

3.1 Overview of an Enterprise Application using 2fa API



1. End-user goes to your web page and initiates a transaction which requires “Two factor authentication”.
2. Your Enterprise application calls our 2fa “Create Session API” with end-user mobile number.
3. Enterprise Application shows missed call number provided by netCORE admin to the end-user.
4. End-user will then give a missed call to the number provided.
5. If the missed call comes within the stipulated time, netCORE can ping an Enterprises API using “Pingback API” if the enterprise has registered its API with netCORE, or the Enterprise can request status by calling our 2fa “Get Session API”.

3.2 Overview of Request/Response cycle using 2fa API

It is a 6 step process which include 3-step on each side i.e. Enterprise and netCORE.

1. Your Enterprise application creates a signature using OAuth and generates an xml request body as specified by the 2fa API.
2. Upon receiving the request, the 2fa web-service verifies the request using the signature provided in the request body.
3. After verifying that it is a valid request the web-service will perform the Enterprises requested operation.
4. The netCORE 2fa web-service will then create a new signature using OAuth and generate the xml response body as specified by the 2fa API.
5. Upon receiving the response, the Enterprise application will verify the response using the signature provided in the response body.
6. After verifying that it is a valid response the Enterprise application will parse the response.

3.3 What is OAuth?

OAuth is an open protocol that aims to standardize the way desktop and web applications access a user's private data. OAuth provides a mechanism for users to grant access to private data without sharing their private credentials (username/password).

Oauth1 is a widely-used, tested, secure, signature-based protocol. The protocol uses a cryptographic signature, (usually HMAC-SHA1) value that combines the token secret, nonce, and other request based information. The great advantage of OAuth 1 is you never directly pass the token secret across the wire, which completely eliminates the possibility of anyone seeing a password in transit. This is the only of the three protocols that can be safely used without SSL (although you should still use SSL if the data transferred is sensitive). However, this level of security comes with a price: generating and validating signatures can be a complex process. You have to use specific hashing algorithms with a strict set of steps. However, this complexity isn't often an issue anymore as every major programming language has a library to handle this for you.

3.4 Why do we use OAuth?

OAuth has been used for encryption as its libraries are available in multiple programming languages, making it easy for enterprises to adopt.
OAuth consist of library function which can be used to generate consumer signature and other unique keys like nonce.

3.5 Why do we need a signature?

The signature is used to verify the authenticity of every request / response. It contains keys & values known only to the Enterprises & netCORE, as well as keys and values shared in the request / response body.

3.6 How are signatures verified?

The keys & values in the request / response body are mapped to the secured2fakey and secret key present with the recipient. Then the recipient tries to recreate the signature it received. If the signatures match then it is a valid request / response.