Encryption Key for Callbacks Mechanism

General Description

The Cedato macro encryption uses a symmetric key algorithm. It uses an encryption key for the encryption and decryption of the relevant values, while authenticating the received data. For this purpose, use the key generated using the Cedato Dashboard. You can find it under the Network Settings->Encryption Key tab.

The encryption mechanism uses the AES-128 cipher in CBC mode and Base64 encoding to provide delivery via URL.

Ad Start price macro purpose is to provide the actual impression price using a callback macro. It is the first macro to use the encryption mechanism as described below.

Algorithms

Algorithm for Encryption as used by Cedato prior to calling the callback URL:

1. Convert rpm_float value to rpm_string.
2. Generate random 16 bytes string initial vector.
3. Encrypt rpm_string using AES-128 cipher in CBC mode and the initial vector
4. Create ciphered_data using concatenation of the initial vector and the encrypted string.
5. Perform Base64 encoding of the ciphered_data.
6. The resulting text is the Base64 encoding with replacement of “+” by “-”, “=” by “~” and “/” by “_” to ensure safe URL deliverability.

Algorithm for Decryption as should be implemented by the customer to retrieve the RPM value:

1. Replace the characters "-", "_", "~" with "+", "/", "=", respectively.
2. Perform Base64 decoding of the encrypted text (after substitution).
3. Split the string into 2 strings.
   The first 16 bytes are the initialization vector and the rest is the encrypted text.
4. To get final result, perform decryption with cipher AES-128 mode CBC using the initialization vector extracted and the encrypted text part.
5. Parse the resulting string into a float.

Sample Code

PHP

```
<?php

function cedato_rpm_decrypt($text,$key)
{
    // prepare for base64 decode
    $rpm_param_prepared = str_replace(array('-','_','~'), array( '+','/','='),$text);
    $data_base64_decoded = base64_decode($rpm_param_prepared);
    $iv = substr($data_base64_decoded,0,16);
    $rpm_encrypted = substr($data_base64_decoded,16);
    $rpm_decrypted = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $key,
$rpm_encrypted, MCRYPT_MODE_CBC, $iv);
    return floatval(trim($rpm_decrypted));
}

?>
```

Python

```
import base64
from Crypto.Cipher import AES

def cedato_rpm_decrypt( text, key ):
    text = text.replace("-","+").replace("_","/").replace("~","=")
    enc = base64.b64decode(text)
    iv = enc[:16]
    cipher = AES.new(key, AES.MODE_CBC, iv )
    dec = cipher.decrypt( enc[16:] )
    dec = dec.partition(b'\0')[0]
    return float(dec)
```

Ruby

```
require 'openssl'
require 'base64'

rpm_text = rpm_text.gsub('~', '=').gsub('-', '+').gsub('_', '/')
base64decoded = Base64.decode64(rpm_text)
iv = base64decoded[0..16]
encrypted = base64decoded[16..-1]
# and now we prepare to decrypt
d = OpenSSL::Cipher.new('AES-128-CBC')
d.decrypt
# now set the key and iv for the decrypting cipher
# this assumes that the password, salt, and iv are known,
# so then you would be able to generate the key as per above
d.key = key
d.iv  = iv

# decrypt the data!
decrypted = '' << d.update(encrypted) << d.finalp Float(decrypted)
```