IPN (Instant Payment Notification) Receiver

Version 0.2
November, 2025

www.eps.com.bd
info@eps.com.bd

1. Overview

This document provides technical guidance for merchants to integrate with the EPS Payment Gateway IPN system. The IPN system allows your application to receive transaction notifications securely whenever a payment is completed.

Key Points: - IPNs are sent as encrypted payloads for security. - The payload must be decrypted using the Secret Key. - Ensure your system validates, stores, and optionally processes these IPNs immediately.

2. IPN Endpoint

URL Example: POST https://merchant-domain.com/api/IPNReceive

Content-Type: application/json

Request Body Example: { “Data”: “IV:EncryptedPayloadBase64” }

3. Payload Structure (Encrypted)

The encrypted payload contains the following fields after decryption:

Field	Type	Description
EpsTransactionId	string	Transaction ID generated by EPS gateway
MerchantTransactionId	string	Merchant’s original transaction ID
StoreId	int	Store identifier
Status	string	Transaction status (e.g., “SUCCESS”)
TotalAmount	decimal	Total transaction amount

Field	Type	Description
StoreAmount	decimal	Amount allocated to the store
TransactionDate	DateTime	Transaction creation date
TransactionType	string	Type of transaction (e.g., CARD, Bkash)
FinancialEntity	string	Payment processor or bank name
CustomerId	string	Customer identifier in merchant system
CustomerName	string	Customer name
CustomerPhone	string	Customer phone number
CustomerEmail	string	Customer email address
CustomerAddress	string	Customer address
Timestamp	long	Unix timestamp of when the IPN was generated

4. Security & Encryption

- The data field in the request body is AES-256 encrypted using the merchant's SecretKey.
- EPS uses CBC mode with PKCS7 padding.
- Each payload contains a unique IV, prepended to the ciphertext (format: `IV:CipherText`).

Decryption Example (C#):

```
string decryptedJson = Decrypt(payload.Data, secretKey);
```

5. Sample Decrypt Method (C#)

```
public static string Decrypt(string encryptedData, string key)
{
    var parts = encryptedData.Split(':');
    byte[] iv = Convert.FromBase64String(parts[0]);
    byte[] cipher = Convert.FromBase64String(parts[1]);

    using var aes = Aes.Create();
    byte[] keyBytes = Encoding.UTF8.GetBytes(key);
    if (keyBytes.Length > 32) keyBytes = keyBytes.Take(32).ToArray();
    else if (keyBytes.Length < 32) keyBytes = keyBytes.Concat(new byte[32 - ke
```

```
yBytes.Length]).ToArray();

    aes.Key = keyBytes;
    aes.IV = iv;
    aes.Mode = CipherMode.CBC;
    aes.Padding = PaddingMode.PKCS7;

    using var decryptor = aes.CreateDecryptor(aes.Key, aes.IV);
    using var ms = new MemoryStream(cipher);
    using var cs = new CryptoStream(ms, decryptor, CryptoStreamMode.Read);
    using var sr = new StreamReader(cs);

    return sr.ReadToEnd();
}
```

5.1 Sample Decrypt Method (PHP)

```
//Decrypt function (AES-256-CBC)
function decrypt($encryptedData, $key)
{
    $parts = explode(':', $encryptedData);
    if (count($parts) != 2) {
        throw new Exception("Invalid encrypted data format");
    }
    $iv = base64_decode($parts[0]);
    $cipherText = base64_decode($parts[1]);

    // Ensure 32-byte key
    if (strlen($key) > 32) {
        $key = substr($key, 0, 32);
    } elseif (strlen($key) < 32) {
        $key = str_pad($key, 32, "\0");
    }

    $decrypted = openssl_decrypt($cipherText, 'AES-256-CBC', $key,
OPENSSL_RAW_DATA, $iv);
    if ($decrypted === false) {
        throw new Exception("Decryption failed");
    }
    return $decrypted;
}
```

6. Handling IPN

1. Receive the POST request at your `/api/IPNReceive` endpoint.
2. Decrypt the `data` field using the SecretKey.
3. Deserialize JSON into your local IPN model.

4. Optional Validations:
    - Check timestamp (replay protection)
    - Check for duplicate transactions
    - Verify transaction amounts and status
5. Store the IPN in your database for record keeping.
6. Return HTTP 200 OK with a JSON message:

```
{
  "status": "OK",
  "message": "IPN received and saved successfully"
}
```

7. Error Handling

- **Invalid Payload**: HTTP 400 Bad Request

```
{
  "status": "ERROR",
  "message": "Invalid payload"
}
```

- **Decryption / Unexpected Errors**: HTTP 500 Internal Server Error

```
{
  "status": "ERROR",
  "message": "Decryption failed or internal error"
}
```

8. Best Practices

- Always validate SecretKey matches what EPS provided.
- Store IPNs in a separate audit table for reconciliation.
- Ensure your endpoint is HTTPS.
- Consider logging errors and failed attempts.

9. Example IPN Model (C#)

```
public class IPNModel
{
    public string EpsTransactionId { get; set; }
    public string MerchantTransactionId { get; set; }
    public string StoreId { get; set; }
    public string Status { get; set; }
    public decimal TotalAmount { get; set; }
    public decimal StoreAmount { get; set; }
    public DateTime TransactionDate { get; set; }
    public string TransactionType { get; set; }
    public string FinancialEntity { get; set; }
```

```
    public string CustomerId { get; set; }
    public string CustomerName { get; set; }
    public string CustomerPhone { get; set; }
    public string CustomerEmail { get; set; }
    public string CustomerAddress { get; set; }
    public long Timestamp { get; set; }
}
```

10. Sample Encrypted POST Request

```
POST https://merchant-domain.com/api/IPNReceive
Content-Type: application/json
{
  "Data": "base64IV:base64CipherText"
}
```