Technical integration

AXEPTA API

AXEPTA is a payment solution based on a **HTTPS POST / GET API** using parameters in the **NVP (Name-Value-Pairs)** method and **JSON objects**.

HMAC authentication and the use of **Blowfish encryption** are used to secure data exchanges.

This section covers the following topics

Integration and inputs for developers

The section Platform Integration for developers gathers all inputs needed to implement Axepta Online in your website

Integration and inputs for developers
Programming basics : Technical implementation Create an API call and use Blowfish encryption HMAC Authentication (Request) HMAC Authentication (Notify) Create a payment request step by step Status inquiries Implementation Sources XSLT Template (.zip) Third-party-cookies - Browser cookies and session handling Technical Integration Guide for PSP

The section Axepta data list gathers the parameters / data used on the platform : error codes, 3DS data, JSON objects

Axepta data
Definitions and abbreviations Currency table A2 Country codes EN A3 AVS match parameters EN A4 Response codes 3DS Data JSON objects list Parameter

Security

HMAC Authentication

To protect you against unauthorized manipulation of your payment transactions, Axepta platform checks with a hash message authentication code (HMAC) if your payment request is authentic and has not been modified. To do so, you transfer a HMAC value to the Platform for each transaction in the MAC parameter.

The Axepta platform uses a Hash Message Authentication Code (HMAC) to verify the authenticity of your payments. MAC SHA-256 algorithm is used with a 32-digit (256-bit) key length.

Further details : HMAC Authentication (Request) et HMAC Authentication (Notify)

Blowfish ECB

Blowfish is a symmetric encryption algorithm (i.e., "secret key").

To ease your integration, you will find below some examples of Blowfish ECB library.

Techno	Examples
ASP	txmsCrypto.dll // txmsCrypto.BlowFish
ASP.NET	Computop.Core.Crypto.dll // Axepta.Core.Crypto.BlowFish
Java	Blowfish.java
PHP	function.inc.php • ctHMAC • ctEncrypt • ctDecrypt

Build a payment request

Concepts

Axepta payment solution integration is based on a payment request build with :

- Parameters using the NVP (Name-Value-Pairs) method
- JSON objects
- Calculation of a HMAC
- A correct character string contains three basic parameters: MID (Merchant ID), Len (Length) and Data (Data). MID and Len parameters are not encrypted. Only the parameter Data is encrypted with the Blowfish method

Parameters

- Parameter **Data** (Données) contains the key data of the payment such as amount and currency
- Parameter **Len** (Longueur) is very important for encryption because it contains the length of the unencrypted string in the Data parameter. The total amount of encrypted data is multiplied by 8 with the Blowfish encryption, so the correct (real) length of the string of characters must be known for decryption, otherwise other characters (not initially added) will appear at the end of the string.

The parameters are transmitted via **HTTPS POST** or **HTTPS GET**.

The recommended transmit method is **HTTPS POST** because the parameter character string in the case of GET is attached to the URL, which is limited to 2048 bytes depending on the browser.

Create a payment request step by step

The steps are :

- Calculate HMAC to secure amount and currency - cf. HMAC Authentication (Request)
- Build and encode JSON objects in Base64 with padding - cf. Payment Features
- Assemble API parameters (key / value, JSON objects)
- Encryption of all API parameters with the Blowfish key: this will provide the **Data** and **Len** parameters
- If necessary, add simple parameters to customize the payment page hosted by (e.g. language="en" to use the English language, customFields)
- Sending the API application to the selected endpoint

Response

Axepta Online uses the POST and GET methods to redirect the buyer to the merchant's e-commerce site or send the payment notification

Payment method	Reponses
Card payments	POST for l'URLFailure / URLSuccess / URLNotify GET for l'URLFailure / URLSuccess / URLNotify fallback 3DSV1
Other payment methods	GET for l'URLFailure / URLSuccess / URLNotify

Example

The page Create a payment request step by step describes the step to create a payment with BNP_DEMO_AXEPTA.

This can be the first step in your integration then you will use your own MID