API Security Handbook

HMAC Request Signing & Authentication Specification

Document Version: 1.1
Audience: Client Engineering & Security Teams
Purpose: Define the mandatory authentication mechanism for secure API access.

1. Introduction
This handbook defines the HMAC-based request authentication mechanism required to securely consume the API. All client integrations must strictly comply with this specification.

2. Security Overview
The API uses HMAC SHA-256 request signing to ensure request authenticity, integrity, and replay attack protection.

3. Credential Management
Clients are issued a confidential Secret Key used exclusively for signing requests. The Secret Key must never be exposed in frontend environments.

4. Mandatory HTTP Headers
REQUESTID, X-TIMESTAMP, X-NONCE, X-SIGNATURE, Content-Type (application/json).

5. Timestamp Specification
UNIX timestamp in seconds. Requests outside the permitted window are rejected.

6. Nonce Specification
A unique random value per request to prevent replay attacks.

7. Request Body Handling
Request body must be valid JSON and serialized exactly as sent.

8. Canonical String Construction
HTTP_METHOD, REQUEST_PATH, TIMESTAMP, NONCE, REQUEST_BODY separated by newlines.

9. Signature Generation
HMAC SHA-256 using the Secret Key, output as hexadecimal string.

10. Request Submission Guidelines
Signed method, path, body, and headers must match the request sent.

11. Server-Side Validation
Timestamp validation, nonce validation, canonical rebuild, HMAC verification.

12. Compliance
Non-compliant requests will be rejected.

```
}
```

JavaScript HMAC Request Signing Guide

Backend Reference Implementation

This section provides the JavaScript (Node.js) reference implementation for generating HMAC SHA-256 signed requests as per the API Security Handbook. This logic must be implemented only on backend systems.

Generate Timestamp

```
function generateTimestamp() {
  return Math.floor(Date.now() / 1000);
}
```

Generate Nonce

```
import crypto from "crypto";

function generateNonce() {
  return crypto.randomBytes(16).toString("hex");
}
```

Prepare Request Body

```
function prepareRequestBody(bodyObject) {
  if (!bodyObject) {
    return "";
  }
  return JSON.stringify(bodyObject);
}
```

Build Canonical String

```
function buildCanonicalString({ method, path, timestamp, nonce, body }) {
  return [
    method.toUpperCase(),
    path,
    timestamp,
    nonce,
    body
  ].join("\n");
}
```

Generate HMAC Signature

```
function generateSignature(secretKey, canonicalString) {
  return crypto
    .createHmac("sha256", secretKey)
    .update(canonicalString)
    .digest("hex");
}
```

Build Request Headers

```
function buildHeaders({ requestId, timestamp, nonce, signature }) {
  return {
    REQUESTID: requestId,
    "X-TIMESTAMP": timestamp.toString(),
    "X-NONCE": nonce,
    "X-SIGNATURE": signature,
    "Content-Type": "application/json"
  };
}
```

End-to-End Example

```
const secretKey = process.env.SECRET_KEY;

const method = "POST";
const path = "/api/v1/redeem";
const requestId = crypto.randomUUID();

const bodyObject = {
  amount: 1000,
  currency: "INR"
};

const timestamp = generateTimestamp();
const nonce = generateNonce();
const body = prepareRequestBody(bodyObject);

const canonicalString = buildCanonicalString({
  method,
  path,
  timestamp,
  nonce,
  body
});

const signature = generateSignature(secretKey, canonicalString);

const headers = buildHeaders({
  requestId,
  timestamp,
  nonce,
  signature
});
```