3. Authentication & HMAC Signing

Server-to-server requests must be signed. Browser widget requests only need a publishable key.

Which authentication do I use?

Endpoint type	Key	Headers
Browser donation/config endpoints	`pk_...`	`X-Partner-Key`
Backend read endpoints	`sk_...` plus HMAC	`X-Partner-Key`, `X-Timestamp`, `X-Signature`
Backend write endpoints	`sk_...` plus HMAC	`X-Partner-Key`, `X-Timestamp`, `X-Signature`

If an endpoint says it requires a secret key, `pk_...` keys are rejected.

Make your first signed request

Set environment variables:

```
export SIR_BASE_URL="https://devapi.sirgiving.org"
export SIR_SECRET_KEY="sk_test_..."
export SIR_HMAC_SECRET="your-hmac-secret"
```

Call a read endpoint:

```
TS=$(date +%s)
PATH_="/v1/partner/users"
BODY=""
BODY_HASH=$(printf '%s' "$BODY" | shasum -a 256 | awk '{print $1}')
SIG=$(printf '%s' "${TS}GET${PATH_}${BODY_HASH}" \
  | openssl dgst -sha256 -hmac "$SIR_HMAC_SECRET" -hex \
  | awk '{print $2}')

curl "$SIR_BASE_URL$PATH_" \
```

```
  -H "X-Partner-Key: $SIR_SECRET_KEY" \
  -H "X-Timestamp: $TS" \
  -H "X-Signature: $SIG"
```

If the request succeeds, your key, timestamp, and signature are valid.

Signature algorithm

The server computes the same value and compares it to `X-Signature`.

```
bodyHash      = SHA256_hex(rawRequestBody)
signedPayload = timestamp + METHOD + pathWithQueryString + bodyHash
signature     = HMAC_SHA256_hex(hmacSecret, signedPayload)
```

Rules:

- `timestamp` is the same string sent in `X-Timestamp`.
- `METHOD` is uppercase, such as `GET` or `POST`.
- `pathWithQueryString` includes the query string.
- `rawRequestBody` must be the exact bytes sent over HTTP.
- Empty bodies use the SHA-256 hash of the empty string.
- The timestamp must be within 5 minutes of server time.

Node.js helper

```
import crypto from 'crypto';

export function signSirRequest(method: string, path: string, body?: unknown) {
  const timestamp = Math.floor(Date.now() / 1000).toString();
  const rawBody = body === undefined ? '' : JSON.stringify(body);
  const bodyHash = crypto.createHash('sha256').update(rawBody).digest('hex');
  const signedPayload = `${timestamp}${method.toUpperCase()}${path}${bodyHash}`;
  const signature = crypto
    .createHmac('sha256', process.env.SIR_HMAC_SECRET!)
    .update(signedPayload)
    .digest('hex');

  return {
    rawBody,
```

```
    headers: {
      'X-Partner-Key': process.env.SIR_SECRET_KEY!,
      'X-Timestamp': timestamp,
      'X-Signature': signature,
      'Content-Type': 'application/json',
    },
  };
}
```

Use the exact `rawBody` string returned by the signing function as the request body. Do not let your HTTP client re-serialize it after signing.

Common errors

Error	Meaning	Fix
`INVALID_API_KEY`	Missing, malformed, inactive, expired, or wrong key	Check the key value and environment
`TIMESTAMP_EXPIRED`	Timestamp missing or outside the 5-minute window	Sync server time and regenerate signature
`INVALID_SIGNATURE`	HMAC did not match	Check path, query string, body bytes, and HMAC secret
`PARTNER_NOT_ACTIVE`	Partner status is not active	Check partner approval/status
`PARTNER_SUSPENDED`	Partner is suspended	Contact SIR Giving support

Idempotency

Action submissions require an `idempotencyKey`. Use a stable key for the real-world operation, such as `order_98765` or `volunteer_shift_abc123`.

If a network call times out, retry with the same `idempotencyKey`. SIR Giving returns the original result instead of issuing duplicate rewards.

Revision #4
Created 2026-05-10 09:11:13 UTC by krtin shet
Updated 2026-05-30 13:31:58 UTC by krtin shet