Brand Engine

API documentation

Generated 2026-08-24 - Hub origin: https://thebrandengine.ai

This document describes the public HTTP API for the Brand Engine hub: authentication, the brand bundle contract, webhook dispatch, and the multi-brand model used by federated tools.

Always-current HTML reference: https://thebrandengine.ai/developers/reference. Machine-readable: https://thebrandengine.ai/api/public/v1/docs/openapi.json

Endpoints

v0 public API

Bearer API key or federation token. Use these for your own services, webhooks, and automations.

GET `/api/public/tools`

Auth: API key (Bearer be_...)

List tools available in the ecosystem, with this brand's entitlement state.

Returns every tool registered on the hub plus whether the calling brand is entitled to it. Use it to render a tool launcher inside a child app.

- Authorization (header, string, required): Bearer <api key>. Brand-scoped.

```
curl https://thebrandengine.ai/api/public/tools \
-H "Authorization: Bearer be_xxxxxxxxxxxxxxx"
```

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

GET `/api/public/brands`

Auth: API key (Bearer be_...)

List brands the API key can access. For brand-scoped keys this returns the one brand.

Also accepts a signed-in user session token (JWT), in which case it returns every brand the user is a member of together with their role.

- Authorization (header, string, required): Bearer <api key> or Bearer <user JWT>.

```
curl https://thebrandengine.ai/api/public/brands \
-H "Authorization: Bearer be_xxxxxxxxxxxxxxx"
```

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

GET `/api/public/brands/me`

Auth: Federation token (Bearer JWT)

List every brand the signed-in user is a member of. Used by children to render a brand switcher.

- Authorization (header, string, required): Bearer <user session token>.

```
curl https://thebrandengine.ai/api/public/brands/me \
-H "Authorization: Bearer <user session token>"
```

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

GET `/api/public/brand/me`

Auth: Federation token (Bearer JWT)

Active brand identity. Honors X-Brand-Id when the caller is a member of the requested brand.

Returns the compiled brand identity for the caller's active brand. The response carries both brand_identity (current) and brand_book (deprecated alias) so older children keep working.

- Authorization (header, string, required): Bearer <user session token>.

- X-Brand-Id (header, uuid): Target a specific brand the user is a member of. Defaults to the active brand.

```
curl https://thebrandengine.ai/api/public/brand/me \
-H "Authorization: Bearer <user session token>" \
-H "X-Brand-Id: 8f1c..."
```

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff. 403 Caller is not a member of the requested brand.

POST `/api/public/events`

Auth: API key (Bearer be_...)

Emit an orchestration event other tools can react to.

Loose v0 ingestion kept for backwards compatibility. New integrations should use POST /api/public/v1/events, which is schema-locked and idempotent.

- Authorization (header, string, required): Bearer <api key>.
- Content-Type (header, string, required): application/json

```
curl -X POST https://thebrandengine.ai/api/public/events \
-H "Authorization: Bearer be_xxxxxxxxxxxxxxx" \
-H "Content-Type: application/json" \
-d '{"type":"campaign.sent","payload":{"campaign_id":"c_123"}}'
```

Status: 200 Accepted. 400 Body failed validation. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

POST `/api/public/federation/mint`

Auth: Federation token (Bearer JWT)

Mint a short-lived federation token. Optional { brand_id } body to target a specific brand.

Called from the hub session (or with a user session token). The returned HS256 JWT lives ~5 minutes and is what a child presents to /federation/exchange.

- Authorization (header, string, required): Bearer <user session token>.

```
curl -X POST https://thebrandengine.ai/api/public/federation/mint \
-H "Authorization: Bearer <user session token>" \
-H "Content-Type: application/json" \
-d '{"brand_id":"8f1c..."}'
```

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

POST `/api/public/federation/exchange`

Auth: Public

Exchange a federation token for the user's brand context.

Public because the token itself is the credential. Verify the returned brand id against your own cache key before serving data.

```
curl -X POST https://thebrandengine.ai/api/public/federation/exchange \
-H "Content-Type: application/json" \
-d '{"token":"eyJhbGciOiJIUzI1NiJ9..."}'
```

Status: 200 Token valid. 401 Token expired, malformed, or badly signed. 500 Unexpected hub error.

v1 ecosystem contract

Server-to-server endpoints used by federated tools. Auth varies per endpoint.

GET `/api/public/v1/brand/by-id/:id/bundle`

Auth: Child app token (CORE_CHILD_APP_TOKEN__<TOOL>)

Server-to-server brand bundle. Pass ?since=<version> for 304-style not_modified.

The canonical read for children. Call it from your backend only - never from the browser. Cache the result in a brand_cache table keyed by brand_id and store identity_version alongside it.

- id (path, uuid, required): Brand id.
- since (query, integer): The identity_version you already have cached. When it matches, the hub returns { not_modified: true } instead of the full bundle.
- Authorization (header, string, required): Bearer <CORE_CHILD_APP_TOKEN__TOOL>.

- X-Tool-Slug (header, string, required): The child tool slug the token belongs to, e.g. dialog-center.

```
curl "https://thebrandengine.ai/api/public/v1/brand/by-id/8f1c.../bundle?since=12" \
-H "Authorization: Bearer $CORE_CHILD_APP_TOKEN__DIALOG_CENTER" \
-H "X-Tool-Slug: dialog-center"
```

Status: 200 Bundle returned (or { not_modified: true }). 401 Bad child token or tool slug. 403 Brand is not entitled to this tool. 404 Brand not found. 500 Unexpected hub error.

POST `/api/public/v1/events`

Auth: Child app token (CORE_CHILD_APP_TOKEN__<TOOL>)

Locked schema, idempotent event ingestion.

Send an idempotency_key with every event. Re-sending the same key is a no-op, so retries after a network blip never duplicate.

- Authorization (header, string, required): Bearer <CORE_CHILD_APP_TOKEN__TOOL>.
- X-Tool-Slug (header, string, required): Child tool slug.
- Content-Type (header, string, required): application/json

```
curl -X POST https://thebrandengine.ai/api/public/v1/events \
-H "Authorization: Bearer $CORE_CHILD_APP_TOKEN__DIALOG_CENTER" \
-H "X-Tool-Slug: dialog-center" \
-H "Content-Type: application/json" \
-d
'{"brand_id":"8f1c...","type":"message.delivered","occurred_at":"2026-07-29T10:00:00.000Z","idempotency_
key":"msg_9931:delivered","payload":{}}'
```

Status: 200 Accepted (or deduplicated). 400 Body failed schema validation. 401 Bad child token. 403 Brand is not entitled to this tool. 500 Unexpected hub error.

POST `/api/public/v1/internal/dispatch-tick`

Auth: WEBHOOK_DISPATCH_SECRET (hub-internal)

Cron-driven webhook fan-out tick.

Hub-internal. Driven by a scheduled job every minute; drains pending child_context_deliveries with exponential backoff. Documented for completeness - child projects never call it.

- Authorization (header, string, required): Bearer <WEBHOOK_DISPATCH_SECRET>. Hub-only secret.

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

POST `/api/public/v1/internal/register-subscription`

Auth: WEBHOOK_DISPATCH_SECRET (hub-internal)

Admin upsert of (brand_id, tool_slug) -> webhook_url.

Hub-internal. Normally handled automatically by the entitlement trigger using the tool's default webhook URL.

- Authorization (header, string, required): Bearer <WEBHOOK_DISPATCH_SECRET>.

Status: 200 Success. 401 Missing, malformed, or revoked credential. 500 Unexpected hub error. Safe to retry with backoff.

GET `/api/public/v1/docs/openapi.json`

Auth: Public

Machine-readable OpenAPI 3.1 description of this API.

Generated from the same source as this page. Point your codegen, agent, or API client at it instead of scraping the docs.

```
curl https://thebrandengine.ai/api/public/v1/docs/openapi.json
```

Status: 200 OpenAPI document.

GET `/api/public/v1/docs/api.pdf`

Auth: Public

Download the current API documentation as a PDF.

Secondary export. The HTML reference at /developers/reference is the primary, always-current format.

```
curl -O https://thebrandengine.ai/api/public/v1/docs/api.pdf
```

Status: 200 PDF document.

Error envelope

Every failure uses this shape. Branch on code, never on message.

```
{
"error": {
"code": "unauthorized",
"message": "Missing or invalid API key"
}
}
```

bad_request (400) - The body or query failed validation.
unauthorized (401) - Missing, malformed, expired, or revoked credential.
forbidden (403) - Authenticated, but not allowed to touch this brand or tool.
not_found (404) - The brand, tool, or resource does not exist.
rate_limited (429) - Too many requests. Honour the Retry-After header.
server_error (500) - Unexpected hub error. Retry with exponential backoff.

Quickstart

From zero to a cached brand bundle in four steps.

1. Get a credential. For your own back-office services, issue an API key (be_...) from the Developer page inside the hub. For a federated child tool, ask the hub owner for CORE_CHILD_APP_TOKEN__<TOOL> and CHILD_WEBHOOK_SECRET__<TOOL>.

2. Confirm access. Call GET /api/public/brands with the API key. You should get back the brand the key is scoped to, including its current identity_version.

3. Read the identity. From your backend call GET /api/public/v1/brand/by-id/:id/bundle with the child token and X-Tool-Slug header. Store the response in a brand_cache row keyed by brand_id, together with identity_version.

4. Stay fresh. Expose a webhook endpoint, verify X-Hub-Signature with CHILD_WEBHOOK_SECRET__<TOOL>, and mark the matching brand_cache row stale on identity.updated. Re-fetch the bundle with ?since=<cached version> on the next read.

Authentication model

Three credential types, three call sites.

API key (be_...) - issued per brand from the API keys tab. Use for your own services calling /api/public/* v0 endpoints. Always brand-scoped.

Federation token - short-lived HS256 JWT minted from a signed-in user's session via POST /api/public/federation/mint. Children present it to /api/public/federation/exchange to receive the user's brand context. Pass { brand_id } in the mint body to target a specific brand (recommended in multi-brand setups).

Child app token (CORE_CHILD_APP_TOKEN__<TOOL>) - long-lived secret used by a child's backend to call the v1 contract (bundle, events). Paired with the X-Tool-Slug header.

Every credential travels in the Authorization header as a Bearer token. Never place a credential in a query string, and never ship a child token or API key to the browser.

Errors and status codes

One envelope for every failure.

Every error response uses the same shape: { "error": { "code": "...", "message": "..." } }. Branch on code, never on the message text - messages are human-facing and may change.

Codes: bad_request (400), unauthorized (401), forbidden (403), not_found (404), rate_limited (429), server_error (500).

Retry policy: retry 429 after the Retry-After header, retry 5xx with exponential backoff capped at 30 seconds and at most 5 attempts, and never retry other 4xx responses - fix the request instead.

Successful responses keep their endpoint-specific shape; the envelope applies to failures only.

Rate limits and batching

How to poll and push without melting anything.

Default budget is 60 requests per minute per credential, with at most 2 concurrent requests. Exceeding it returns 429 with Retry-After.

Never poll the bundle endpoint on a timer without ?since=<cached version>. With the version parameter a no-change read is a cheap { not_modified: true }; without it you pull the full identity every cycle.

Event ingestion is per-event and idempotent. Batch client-side into 200-500 event chunks, send them sequentially, and always include an idempotency_key so a retry after a network blip cannot duplicate.

Persist your cursor or last-seen version only after your own transaction commits. A failed batch must leave the cursor where it was.

Prefer push over poll: subscribe to the identity.updated webhook and let it invalidate your cache instead of scheduling frequent reads.

Brand bundle contract

How children fetch and cache identity.

Children call GET /api/public/v1/brand/by-id/:id/bundle?since=<version> from their backend. When since matches the current identity_version, the hub returns { not_modified: true } - the child clears its stale flag and serves the cached bundle.

On a 200 the body matches BrandBundleSchema. Children upsert into a brand_cache(brand_id PK, identity_version, bundle, stale, fetched_at) table - ONE ROW PER BRAND. Never collapse to a singleton.

The bundle carries two top-level design blocks alongside the identity document. colors = { source, palette[] } where every entry has id, name, role (primary | secondary | accent | neutral | background |

text | success | warning | danger), hex (canonical, lowercase #rrggbb), rgb { r, g, b } (derived), usage and locked. Locked entries are admin overrides and never change on regeneration - treat hex as the source of truth and derive any other colour space yourself.

logos = { items[], rules[] }. Each item has variant (logo_positive | logo_negative | symbol | wordmark | favicon | other), label, usage, background, clear_space, min_width_px, asset_id and a signed url (30 min TTL) plus mime. Re-fetch the bundle rather than hot-linking an expired URL, and pick logo_negative for dark surfaces.

Webhook dispatch

How identity changes propagate to children.

Editing a brand's identity bumps identity_version and enqueues a delivery in child_context_deliveries for every matching child_context_subscriptions row. A scheduled job hits POST /api/public/v1/internal/dispatch-tick every minute; the hub POSTs identity.updated to each child with X-Hub-Signature (HMAC-SHA256 over the raw body, secret per child).

Children verify the signature with CHILD_WEBHOOK_SECRET__<TOOL>, mark the matching brand_cache row stale = true, and return 200 fast. The next bundle read refreshes the cache.

Deliveries retry with exponential backoff. Your endpoint must be idempotent: the same identity.updated may arrive more than once.

Multi-brand model

One account, many brands.

A user can be a member of multiple brands (memberships table). Each user has one active brand at a time, persisted in user_active_brand. The hub mirrors the active brand slug to a .thebrandengine.ai cookie (be_active_brand) so every child subdomain knows on boot which brand the user is on.

Child responsibility: read the be_active_brand cookie, mint a federation token with { brand_id }, and key every cache + query by that brand id. When the cookie changes (focus event or BroadcastChannel), re-mint and refetch.

Subscriptions are keyed (brand_id, tool_slug) - a new brand gaining an entitlement auto-creates its subscription via DB trigger, so identity webhooks flow without manual setup.

Where to find a Brand ID: sign in to the hub and open Developer -> API keys -> Brands & IDs. Every brand you are a member of is listed there with its UUID and slug, with a copy button. The active brand's ID is also shown in the Brand page header.

Pairing keys with brands: an API key row stores brand_id, so a key only ever resolves one brand. To connect a second brand to the same tool, switch brands in the sidebar switcher, issue a new key, and send that key with the matching Brand ID (or the X-Brand-Id header where the endpoint accepts it).

Listing brands: from a child, call GET /api/public/brands/me with the user's federation token to render a brand switcher; from a back-office integration, call GET /api/public/brands with the API key (returns the single brand the key is scoped to).

Adoption checklist

For child projects integrating the contract.

Use the brand-ecosystem-contract skill from the child project's chat, or follow the README at src/lib/ecosystem-contract/v1/README.md.

Copy the schemas and webhook helpers verbatim - they must stay byte-identical to the hub.

Machine-readable contract: GET /api/public/v1/docs/openapi.json. Human reference: /developers/reference. PDF export: /api/public/v1/docs/api.pdf.