Use the DHI API

- Endpoint
- Request format
- Authentication
- Response format
- Queries
  - imagePackagesForImageCoords
    - Response fields
    - Mapping to OpenVEX

The DHI API is a GraphQL API for querying Docker Hardened Images data programmatically, for use cases like building automation or dashboards on top of DHI data.

Endpoint

Send requests as `POST` requests to:

```
https://api.dso.docker.com/v1/graphql
```

Request format

The API accepts standard GraphQL requests: a JSON body with a `query` and, optionally, `variables`.

```
$ curl https://api.dso.docker.com/v1/graphql \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/json" \
  -d '{"query": "...", "variables": { ... }}'
```

Every query takes a `Context` argument (conventionally named `ctx` in the `variables` object) alongside its query-specific arguments:

Argument	Type	Required	Description
`ctx`	`Context`	Yes	Scopes the request to an organization.
`ctx.organization`	`String`	Yes	The Docker organization the token belongs to.

Authentication

An organization access token (OAT) or personal access token (PAT) isn't used directly as the bearer token. Exchange it first for an access token:

```
$ curl -X POST https://hub.docker.com/v2/auth/token \
  -H "Content-Type: application/json" \
  -d '{"identifier": "<identifier>", "secret": "<token>"}'
```

For `identifier`, use your Docker Hub username with a PAT, or the organization name with an OAT. The response contains the access token:

```
{ "access_token": "..." }
```

Pass that `access_token` as `Authorization: Bearer <access_token>`. Also set `ctx.organization` in `variables` to the organization the token belongs to (see Request format).

Response format

Responses follow the standard GraphQL envelope:

Key	Description
`data`	The requested fields. A field is `null` if it couldn't be resolved, for example due to an authorization failure.
`errors`	Present when a field failed to resolve. Includes a `message` and a `path` identifying which field failed.
`extensions`	Metadata such as a `correlation_id`, useful when reporting an issue.

For example, an unauthenticated request, or a request for data your token can't access, returns a `null` result under `data` alongside an authorization error in `errors`, rather than an HTTP-level failure:

```
{
  "errors": [
    {
      "message": "You are not allowed to read data for this team",
      "path": ["someQuery"],
      "extensions": { "code": "DOWNSTREAM_SERVICE_ERROR", "status": 403 }
    }
  ],
  "data": { "someQuery": null },
  "extensions": { "correlation_id": "..." }
}
```

Queries

`imagePackagesForImageCoords`

Fetches every package in an image, every CVE reported against it, and whether Docker suppresses that CVE, by digest. See Query VEX for a Docker Hardened Image for a guided example.

Argument	Type	Required	Description
`digest`	`String`	Yes	The image's platform manifest digest, not the multi-arch index digest.
`hostName`	`String`	Yes	`hub.docker.com` or `docker.io`.
`repoName`	`String`	Yes	Repository name, with or without the namespace prefix.
`includeExcepted`	`Boolean`	No	Include suppressed CVEs in the response alongside the reason for suppression. Without it, the response only shows the netted list, with no visibility into what was suppressed.
`includeNodsa`	`Boolean`	No	Include Debian NODSA exclusions, which make up most suppressions on a Debian-based image.

Argument	Type	Required	Description
`includePublic`	`Boolean`	No	Also include public images when `ctx.organization` scopes the request to an organization. Not needed for a typical lookup.

Keep the requested response fields limited to what you plan to render. Fields such as `locations`, `description`, `vulnerableRange`, and `epss` increase response size substantially and aren't needed for a CVE-count or suppressed-CVE view.

Response fields

`vulnerabilityExceptions` only contains records that actually suppress a CVE, so it always lines up with `isExcepted`: an empty array means the CVE is live. Use `isExcepted` as your filter for "is this CVE suppressed."

Field	Meaning
`isExcepted`	Docker suppresses this CVE for this image. Use this to filter.
`sourceType`	`EXTERNAL` (Debian NODSA), `MANUAL_EXCEPTION` (Docker analyst exception), or `VEX_STATEMENT` (an ingested VEX document).
`type`	`FALSE_POSITIVE` and `ACCEPTED_RISK` suppress the CVE. `UNDER_INVESTIGATION` and `AFFECTED` don't.
`justification`	The OpenVEX justification value. Always `null` for NODSA exclusions.
`additionalDetails`	Free-text rationale for the suppression.
`isDhiStatement`	Whether the statement is inherited from the DHI base image.
`id`	Stable identifier for the statement.

Mapping to OpenVEX

If your pipeline consumes OpenVEX documents (for example, Trivy's `--vex` flag), each suppressed record maps as follows:

OpenVEX field	Source
`vulnerability.name`	`sourceId`
`products[].@id`	The parent package's `purl`
`status`	`not_affected` (from `type: FALSE_POSITIVE`)
`justification`	`justification`, defaulting to `vulnerable_code_cannot_be_controlled_by_adversary` for NODSA exclusions

OpenVEX field	Source
`status_notes`	`additionalDetails`
`@id`	`id`

□□□ #2
□ □□□ □□□ 2026-10-31 19:19:00 UTC
□ □□□ □□□ 2026-10-31 19:19:01 UTC