

Build your own connector

From the SourceLace manual: How to connect SourceLace to a system it has no connector for, such as an in-house database, an internal API or a niche app, by building a small web service of your own. This page is the complete protocol: if something you need is not written here, it is not part of the p

SourceLace version 0.1.0 · Made October 4, 2026

The latest version is at <https://sourcelace.onrender.com/docs>

Build your own connector

How to connect SourceLace to a system it has no connector for, such as an in-house database, an internal API or a niche app, by building a small web service of your own. This page is the complete protocol: if something you need is not written here, it is not part of the protocol, so tell support@sourcelace.com and we will add it here.

Status: new. The protocol is version 1 and covered by automated tests; it has not yet been used with a customer's live connector. Admins add one as **Custom connector** on **Manage sources**.

How it works

A connector is a small HTTPS web service that you write and run, in any language. SourceLace calls it when someone asks about your system: "what objects are there", "what fields does this one have", "run this query", "get this record", and, if you allow it, "check this change" and "make this change". Every call is signed with a secret you share with SourceLace, and names the person who is asking, so your connector applies your own permissions.

Your code never runs inside SourceLace, and SourceLace never sends your connector anyone's passwords or tokens: only the signed request and the person's email, name (if known), organization id and SourceLace group names.

SourceLace does:

- signs people in, and checks that the person may use this source at all ([access by group](#));
- previews every change and applies it only after the person confirms it, and only when an admin has turned on changes for this source and object;
- writes every call to the audit trail;
- holds results in memory briefly (never in a database), caps the rows it keeps, and stops waiting for your connector after 30 seconds.

Your connector must:

- check the signature and timestamp of every request (see [Signing requests](#));
- **enforce your own system's permissions**: use person to decide what this person may read and change, and return or change nothing they may not;
- **keep query read-only**: whatever the query text says, /query must not change any data. Run it with a read-only login or connection, or accept only a format you can check. SourceLace only checks that a query is not empty and at most 10,000 characters, because only your connector knows its query language;
- never put passwords, keys or internal addresses in an answer: `error.message` is shown to the person.

Add a custom source (SourceLace admin)

1. Make a shared secret, a random string of **32 to 256 characters**, for example with `openssl rand -hex 32`. Keep it in your password manager and give it to whoever runs the connector.

2. On **Manage sources** → **Add a source**, choose **Custom connector** (custom), give it a name (such as `custom:inventory`) and a label, and fill in the options below.
3. To allow changes, turn on **Allow changes** and list the objects that may change. The connector must also say `"writes": true` in its manifest.
4. People connect the source once (there is no extra sign-in) and then ask about it like any other source.

Option	Type	Default	Example	What it is
url	Web address	(required)	<code>https://connector.corvanta.com</code>	The connector's address. Must start with <code>https://</code> , with no user name, password, <code>?query</code> or <code>#fragment</code> . A path is fine, such as <code>https://api.corvanta.com/source-lace-connector</code> .
secret	Secret, 32 to 256 characters	(required)		The shared secret. Stored encrypted with your organization's key and never shown again.

In SourceLace's cloud, the connector's address must be on the public internet: private addresses are refused.

Transport

Rule	Value
Base URL	The url option. Every path below is appended to it.
Scheme	<code>https://</code> only.
Methods	GET for the manifest; POST for everything else.
Request body	UTF-8 JSON, Content-Type: <code>application/json</code> . SourceLace sends compact JSON, but don't depend on spacing or key order: parse it.
Success	HTTP 200 with a UTF-8 JSON body. Any other status is an error (see Errors).
Redirects	Not followed. A 3xx answer is an error.
Time limit	SourceLace waits at most 30 seconds for the whole answer.
Size limit	An answer must be at most 10 MB (10,000,000 bytes).
Unknown fields	Both sides ignore JSON fields they don't know. Later versions may add optional fields; they will never change or remove the ones here.
Null	A field marked "or null" may be <code>null</code> or left out; both mean "no value".

Signing requests

Every request, including the GET, has two headers:

Header	Value
SourceLace-Timestamp	When SourceLace sent the request: whole seconds since 1970-01-01T00:00:00Z, as digits, such as 1767225600.

Header	Value
SourceLace-Signature	v1= followed by 64 lowercase hexadecimal characters.

SourceLace computes the signature like this, and your connector must check it the same way:

1. Take the raw request body **exactly as the bytes arrived**. Don't parse and re-encode it first. For the GET request the body is empty: zero bytes.
2. Build the message: the timestamp header's value as ASCII bytes, then one . (byte 0x2E), then the body bytes.
3. Compute HMAC-SHA256 of the message, keyed with the secret's UTF-8 bytes.
4. Write the 32-byte result as 64 lowercase hexadecimal characters and put v1= in front.

Refuse the request (HTTP 401, error code unauthorized) when either header is missing, the timestamp is not only digits, the timestamp is more than **300 seconds** before or after your own clock (keep your clock synchronized with NTP), or the signature is not exactly equal to the one you compute. Compare with a constant-time comparison, such as `hmac.compare_digest` in Python, `crypto.timingSafeEqual` in Node.js or `MessageDigest.isEqual` in Java.

To change the secret without downtime, a connector may accept two secrets for a while: add the new one next to the old one in your connector, change it in SourceLace, then remove the old one.

Worked example

Your code must produce exactly these signatures.

Secret	sourceLace-example-secret-0123456789
Timestamp	1767225600
Body (110 bytes, no line break at the end)	{"person":{"email":"ann@corvanta.com","name":"Ann Lee","tenant":"corvanta","groups":["Sales"]},"text":"stock"}
Message that is signed	1767225600. followed by the body
SourceLace-Signature	v1=fefe61f4029cc019764e6d73619e17f4798c43292f94e370244fd17116c27623

For a GET (empty body) with the same secret and timestamp, the message is 1767225600. and the signature is v1=5c4452cf2f35fa29be49a2e8d9efc44409c6dd800b9abf2c17bf0a27a56f8548.

Check it from a shell:

```
printf '%s' '1767225600.{\"person\":{\"email\":\"ann@corvanta.com\",\"name\":\"Ann Lee\",\"tenant\":\"corvanta\",\"groups\":[\"Sales\"]},\"text\":\"stock\"}' \\
| openssl dgst -sha256 -hmac 'sourceLace-example-secret-0123456789'
# ... fefe61f4029cc019764e6d73619e17f4798c43292f94e370244fd17116c27623
```

Endpoints

Every POST body has a person object. SourceLace verified who this is; once the signature checks, you can trust it.

person field	Type	Meaning
email	string, 1-320 characters	The person's email address, in lowercase.
name	string (up to 200) or null	Their display name, or null when SourceLace does not know it.
tenant	string, 1-100 characters	Their organization's id in SourceLace, such as corvanta. One connector can serve several organizations; use this to tell them apart.
groups	array of strings (up to 500)	The person's SourceLace group names (set by an admin, or synced from single sign-on). May be empty.

The lengths below are maximums your answers must respect. SourceLace treats an answer that breaks any rule here as a failed call, and does not use it.

GET /sourcelace/v1/manifest

What the connector is. No body. SourceLace asks for it when someone connects the source, and keeps the answer for 5 minutes.

Field	Type	Required	Meaning
protocol	integer, 1	no (default 1)	The protocol version. Only 1 exists.
name	string, 1-100	yes	The system's name, such as Corvanta inventory.
description	string, 1-1000	yes	What is in it, in a sentence or two.
version	string, 1-50	yes	Your connector's own version, such as 1.0.0.
query_language.name	string matching <code>^[a-z][a-z0-9_]{0,39}\$</code>	yes	A short name, such as inventory_sql.
query_language.description	string, 1-2000	yes	One paragraph that tells the AI how to write a query: the objects or tables, the syntax, and what is refused. Write it for a reader who has never seen your system.
query_language.examples	1 to 3 strings, each 1-1000	yes	Complete example queries.
writes	boolean	no (default false)	true when the two changes endpoints work.
objects_searchable	boolean	no (default false)	true when /objects filters by text itself. Otherwise SourceLace keeps the objects whose name, label or description contains the text.

```
{
  "protocol": 1,
  "name": "Corvanta inventory",
  "description": "Products, stock levels and warehouses from Corvanta's inventory system.",
  "version": "1.0.0",
  "query_language": {
```

```

    "name": "inventory_sql",
    "description": "One SQLite SELECT statement on the tables products and warehouses. Anything
that changes data is refused.",
    "examples": ["SELECT sku, name, in_stock FROM products WHERE in_stock < 100 ORDER BY in_stoc
k"],
  },
  "writes": true,
  "objects_searchable": false
}

```

POST /sourcelace/v1/objects

The objects (tables, entities, record types) this person can read.

- **Request:** person; text, 0-200 characters, what the person is looking for (empty means all).
- **Answer:** objects, at most 1,000, each with name (1-200, required, used in every other call), label (up to 200, or null) and description (up to 1,000, or null).

```

{"objects": [{"name": "products", "label": "Products", "description": "Everything we sell."}]}

```

POST /sourcelace/v1/describe

One object's fields.

- **Request:** person; object, a name from /objects.
- **Answer:** name, optional label and description, and fields: 1 to 1,000 fields with unique names, each with name (1-200), type, optional label and description, key (true for the one field whose value /record takes as id; at most one) and writable (true when a change may set it). If there is no such object, answer not_found.

Type	JSON value in rows
string	string
integer	number without a fraction
number	number
boolean	true or false
date	string, YYYY-MM-DD
datetime	string, ISO 8601 with a time zone, such as 2026-10-03T14:05:00Z
json	any JSON value

Any value may be null.

POST /sourcelace/v1/query

Run one read-only query in your query language.

- **Request:** person; query, 1-10,000 characters; max_rows, 1-10,000. SourceLace asks for one more row than it keeps, to tell whether a result was cut short.
- **Answer:** columns (at most 500 unique names, in display order), rows (at most max_rows objects whose keys are names from columns; a missing key means null) and truncated (true when more rows matched than you returned).
- If the query is not valid, or would change anything, answer invalid_query with a message that says how to fix it: SourceLace passes it to the AI, which uses it to try again.

```
{"columns": ["sku", "in_stock"], "rows": [{"sku": "SKU-2", "in_stock": 0}], "truncated": false}
```

POST /sourcelace/v1/record

One record by the value of its key field.

- **Request:** person; object; id, 1-500 characters (the key field's value, as text).
- **Answer:** row, an object of field name to value. If there is no such object or record, or the person may not see it, answer `not_found`.

```
{"row": {"sku": "SKU-1", "name": "Trail running shoe", "price": 129.0}}
```

Changes (optional): /changes/check and /changes/apply

POST /sourcelace/v1/changes/check and POST /sourcelace/v1/changes/apply, only when the manifest says "writes": true. SourceLace calls check to build the preview the person sees, and apply only after they confirm it.

A change, in both requests next to person, has operation (create, update or delete), object, record_id (null for create) and fields (at most 100 new values; {} for delete, at least one otherwise).

/changes/check must not write anything. It answers allowed (boolean, required), problems (up to 50 reasons, for the person), before (the record's current values of the fields being changed), record_name (shown in the preview) and version (the record's current version, such as a last-modified time; SourceLace sends it back to apply). If the record does not exist, answer `not_found`; if the person may not change it, `forbidden`, or `allowed: false` with problems.

/changes/apply makes the change. Its request also has version, as check returned it (or null). If the record's version is no longer version, don't write and answer `conflict`. It answers record_id, the record's id (the new id after a create).

```
{"record_id": "SKU-4"}
```

Errors

Any answer other than HTTP 200 is an error. Its body should be:

```
{"error": {"code": "not_found", "message": "There is no product SKU-9."}}
```

message is 1-1,000 characters and **is shown to the person** and their AI, so write it for them and never include secrets.

code	HTTP status	When
invalid_request	400	The request does not follow this protocol (you will mostly see this while testing).
invalid_query	400	The query is not valid in your language, or would change data.
unauthorized	401	The signature or timestamp check failed.
forbidden	403	This person may not do this.
not_found	404	No such object or record (or the person may not know it exists).

code	HTTP status	When
conflict	409	/changes/apply only: the record changed since /changes/check.
unavailable	503 (or 500 for an unexpected failure)	Your system cannot answer right now.

An error answer without this body, a timeout, a refused connection or an invalid answer is reported to the person as a failure of the connector, without details.

Limits

Limit	Value
Time per answer	30 seconds
Size per answer	10 MB
Query text	10,000 characters
Rows per query request	10,000 (and SourceLace keeps at most 2,000)
Objects per /objects answer	1,000
Fields per object	1,000
Columns per query	500
Fields per change	100
Signature clock tolerance	300 seconds
Shared secret	32 to 256 characters

Versioning

This is version 1, and the path contains it (/sourcelace/v1/). Additions that are optional on both sides stay in version 1, so ignore what you don't know. Anything that would break an existing connector will be version 2, under /sourcelace/v2/, and SourceLace will keep calling version 1 connectors.

Building one

Any language that can serve HTTPS and compute HMAC-SHA256 works:

1. Implement signature checking first and test it against the [worked example](#).
2. Implement the manifest and the four POST endpoints (objects, describe, query, record), and the two changes endpoints only if you want changes.
3. Answer errors in the shape above.
4. Run it behind HTTPS on the internet (any container platform, a virtual machine behind a reverse proxy, or your own Kubernetes), with the secret as a secret setting.

SourceLace also has a Python starter kit, `sourcelace-connector` (Python 3.11 or newer): you write one class with objects, describe, query and record methods plus a manifest, and the kit handles signing, validation and errors. Its `sourcelace-connector check <address> --secret <secret>` command calls every endpoint the way SourceLace does and prints a pass or fail list, for a

connector in any language, without changing anything. It is not yet published; ask support@sourcelace.com for a copy.

When something goes wrong

What you see	What to do
"Set the option 'url': the connector's address, such as <code>https://connector.corvanta.com</code> ."	Fill in url.
"The connector's url must start with <code>https://...</code> "	Use the connector's <code>https://</code> address.
"The option 'url' must be a plain address such as <code>https://connector.corvanta.com</code> , with no user name, password, <code>?query</code> or <code>#fragment</code> ."	Remove those parts from url.
"Set the option 'secret': the secret shared with the connector, 32 to 256 characters..."	Enter the shared secret.
"The address of ...'s connector (...) could not be found."	Check the host name in url.
"... is a private network address, and this SourceLace server only calls connectors on the internet..."	Give the connector a public <code>https://</code> address.
"... 's connector refused SourceLace's signature (...). An admin should check that the secret on SourceLace's Sources page matches the connector's, and that the connector's clock is right."	Make the secrets match, and sync the connector's clock.
"... 's connector did not answer within 30 seconds."	Make the connector faster, or ask for less.
"... 's connector sent an answer larger than 10 MB."	Return fewer rows or smaller values.
"... 's connector sent an answer that does not follow the SourceLace connector protocol, so SourceLace did not use it..."	Whoever maintains the connector runs <code>sourcelace-connector</code> check to see what is wrong.
"... 's connector failed (HTTP ...). Try again; if it keeps failing, tell your admin."	The connector answered with an error but no error body. Check its logs.
"... refused the query: ..."	The connector's own reason follows; rewrite the query.
"... does not accept changes: its connector is read-only."	The manifest says <code>"writes": false</code> .