

Any REST or SOAP system, without code

From the SourceLace manual: How to connect a system
SourceLace has no ready-made connector for, when it has a REST API (with or without an OpenAPI document) or a SOAP web service (with a WSDL). An admin describes the API on Manage sources; nobody writes or runs any code. For systems with no web API

SourceLace version 0.1.0 · Made October 7, 2026

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

Any REST or SOAP system, without code

How to connect a system SourceLace has no ready-made connector for, when it has a REST API (with or without an OpenAPI document) or a SOAP web service (with a WSDL). An admin describes the API on **Manage sources**; nobody writes or runs any code. For systems with no web API at all, [build your own connector](#) instead.

Status: Preview. New, offered for pilots and provided as is. Neither kind has been used with a customer's live system yet.

At a glance

	Connects through	Sign-in	Network	Status	First test question
Any REST API	The API's own REST endpoints, from its OpenAPI document or endpoints an admin lists	Personal sign-in with oauth; shared account otherwise	Allow SourceLace's IP addresses, if the API limits them	Preview	Test connection on Manage sources , then "Which operations does rest:billing offer?"
Any SOAP web service	The service's SOAP 1.1 or 1.2 endpoint, from its WSDL	Personal sign-in with oauth; shared account otherwise	Allow SourceLace's IP addresses, if the service limits them	Preview	Test connection , then "Which operations does soap:customers offer?"

Add each source with the [usual steps](#). The network answers are explained in [What your network needs](#). If the first test question fails, the error message tells you what to fix: look it up under **When something goes wrong** below.

How it works

- **REST** (rest): every operation in the API's description becomes an object. People (and their AI) find operations with search, see each one's parameters, and read with a query that names an operation and its parameters, in the rest_json language: {"operation": "listInvoices", "params": {"status": "open"}}. SourceLace checks every query against the description (the operation exists and is a read, required parameters are there, nothing unknown, values have the right type and allowed values) before anything is sent, and pages through the results itself.
- **SOAP** (soap): every operation the admin marks as a read becomes an object, queried the same way in the soap_json language: {"operation": "GetCustomers", "params": {"Country": "DE"}}. SourceLace builds the SOAP 1.1 or 1.2 message, reads the answer, and turns SOAP faults into messages that say what to fix.
- **Only reads, unless you allow more.** For REST, only GET operations are reads. Every other operation (POST, PUT, PATCH, DELETE) is off until an admin switches it on by name. A switched-on operation never runs as a query: it runs only as a change, which the person sees first (the exact request, and the record's current values when the API has a GET on the same path) and confirms. SOAP sources are read-only.
- **Answers are data.** What the system sends back only ever becomes rows; nothing in it is followed as an instruction.

How people sign in

Choice (auth)	Who the system sees	When to use it
oauth	Each person, with their own account (OAuth 2.0 authorization code flow with PKCE)	Recommended whenever the system offers OAuth: its own permissions then apply to each person.
api_key, bearer, basic	One shared account for everyone who uses the source	Only when the system offers nothing per person.
none	Nobody (a public API)	Public data only.

Shared access: with `api_key`, `bearer`, `basic` or `none`, everyone who may use the source acts as the same account, so the system cannot tell people apart and cannot apply its own permissions per person. SourceLace says so wherever the source is listed. Give such a key the least access it needs (read-only if possible), and limit who can use the source with [access by group](#).

Secrets (the OAuth client secret, the API key, token or password) are stored encrypted with your organization's key, shown only as "(saved)", never sent anywhere but the system itself, and never written to logs or answers.

Add a REST source (SourceLace admin)

1. If the system offers OAuth, have its admin register an OAuth app (client) for SourceLace, with exactly this redirect URL, <https://sourcelace.onrender.com/connect/callback>, and only the read scopes people need. Otherwise, create an API key or a service account with the least access needed.
2. On **Manage sources** → **Add a source**, choose **Any REST API, no code** (`rest`), give it a name (such as `rest:billing`) and a label.
3. Paste or load the API's OpenAPI 3 or Swagger 2 document (JSON or YAML), or give its address in `spec_url`: SourceLace fetches it once when you save and keeps it. To refresh it later, empty the document box and save again. Without an OpenAPI document, list a few endpoints by hand (below).
4. Choose how people sign in, and fill in its options.
5. Save, then press **Test connection**: SourceLace checks the description, that the address may be called, and (for shared access) reads a few records.
6. To allow changes, turn on **Allow changes** and name the operations, such as `createInvoice`, `updateInvoice`.

Option	Example	What it is
<code>spec</code>	(an OpenAPI document)	The API's description, JSON or YAML. Up to 3 MB. Other files it refers to are not fetched.
<code>spec_url</code>	https://api.corvanta.com/openapi.json	Where to fetch spec once, when spec is empty.
<code>endpoints</code>	(JSON, below)	Endpoints written by hand; they add to (or replace, by name) the document's operations.

Option	Example	What it is
base_url	https://api.corvanta.com/v1	The API's address, when the document has none, or to use another (such as a sandbox).
auth	oauth	oauth, api_key, bearer, basic or none.
client_id, client_secret		The OAuth client (the secret only for a confidential client).
authorize_url, token_url, scopes	https://auth.corvanta.com/oauth/authorize	OAuth addresses and scopes. Empty: taken from the OpenAPI document's OAuth flow.
secret		The API key, token or password for shared access.
key_name, key_in	X-API-Key, header	For api_key: the header or query parameter that carries it.
username	svc-sourcelace	For basic.
test_operation	listInvoices	The read Test connection uses (default: the first GET without required parameters).

Endpoints by hand look like this (only name and path are required; method defaults to GET):

```
{
  "base_url": "https://api.corvanta.com/v1",
  "endpoints": [
    {
      "name": "invoices",
      "method": "GET",
      "path": "/invoices",
      "description": "Invoices, newest first",
      "params": [
        {
          "name": "status",
          "in": "query",
          "enum": ["open", "paid"]
        }
      ],
      "rows": "data",
      "paging": {
        "style": "cursor",
        "param": "starting_after",
        "next": "meta.next_cursor"
      },
      {
        "name": "invoice",
        "path": "/invoices/{id}"
      },
      {
        "name": "createInvoice",
        "method": "POST",
        "path": "/invoices",
        "body": ["amount", "customer"]
      }
    ]
  ]
}
```

- params: each with name, in (query, path or header), and optionally required, type (string, integer, number, boolean, array), enum and description. Every {name} in the path is a required path parameter.
- rows: where the records are in the answer, as a dotted path (data, result.items); empty for an answer that is itself the list. Left out, SourceLace looks for a list such as data, items or results.
- paging: style is page (with param, such as page), offset (with param, such as offset, and size_param), cursor (with param and next, the dotted path to the next cursor in the answer), link (the Link: <...>; rel="next" header) or none. size_param and size set how many rows a page asks for. For OpenAPI documents SourceLace works this out from the parameter names; an operation can also say it with x-sourcelace-paging and x-sourcelace-rows.
- body: for changes, the JSON body's field names.

Changes

A change names its operation as the object: POST operations are creates, PUT and PATCH updates, DELETE deletes. Its fields are the operation's parameters (by name) and its JSON body fields; the record id fills the path's last parameter (or several, as a JSON object such as {"customerId": "C-1", "id": "7"}). Each change shows the exact request before the person confirms it.

Add a SOAP source (SourceLace admin)

1. Create a service account with read-only access, or an OAuth client if the service supports it.
2. On **Manage sources** → **Add a source**, choose **Any SOAP web service, no code** (soap), with a name such as soap:customers.
3. Paste or load the WSDL, or give its address in `wsdl_url`: SourceLace fetches it once when you save and keeps it (empty the box and save again to refresh it).
4. List the operations that only read in `read_operations`, such as `GetCustomers`, `GetOrders`. Nothing else can run.
5. Choose how people sign in (the same choices and options as REST), save, and press **Test connection**.

Option	Example	What it is
<code>wsdl</code>	(a WSDL document)	The service's WSDL 1.1. Up to 3 MB. Imported files are not fetched: their types show as unknown, and parameters are passed by name.
<code>wsdl_url</code>	<code>https://soap.corvanta.com/customers?wsdl</code>	Where to fetch wsdl once, when it is empty.
<code>read_operations</code>	<code>GetCustomers</code> , <code>GetOrders</code>	The only operations that may run.
<code>endpoint</code>	<code>https://soap.corvanta.com/customers</code>	The service's address, if not the WSDL's.
<code>soap_version</code>	1.2	SOAP 1.1 or 1.2; empty uses the WSDL's (1.1 first).

Supported: document/literal (including the common "wrapped" style) and rpc/literal. SOAP encoding (`use="encoded"`) is not.

Safety

These sources reach addresses your admin types in, so SourceLace guards every request:

- only `https://` (a self-hosted server's operator may allow `http://` with `SOURCELACE_ALLOW_HTTP_API_SOURCES`);
- in SourceLace's cloud, only addresses on the public internet: private, loopback, link-local and other reserved addresses are refused, every address a name resolves to is checked, and the request goes to the address that was checked. Cloud metadata addresses are always refused, on any server;
- redirects are never followed, and next-page links must stay on the API's own address;

- answers are limited to 10 MB, and each query to the organization's query time (`queries.timeout_seconds`) and row cap;
- each source sends at most `queries.api_requests_per_minute` requests a minute (120 by default; see [Limits](#));
- nothing of SourceLace's own (cookies, tokens, keys) is ever sent: only the source's own credential;
- XML (the WSDL and every SOAP answer) is read with DTDs and entities refused, so XML attacks such as external entities and "billion laughs" do not work.

When something goes wrong

What you see	What to do
"Give the source an API description: paste or upload an OpenAPI document (option <code>spec</code>), or list a few endpoints by hand (option <code>endpoints</code>)."	Add the OpenAPI document, <code>spec_url</code> , or endpoints.
"Give the source its WSDL: paste or upload it (option <code>wsdl</code>), or set <code>wsdl_url</code> ."	Add the WSDL, or <code>wsdl_url</code> .
"Fetching the API description from <code>spec_url</code> answered HTTP ..." (or "Fetching the WSDL from <code>wsdl_url</code> ...")	Check the address, or paste the document instead.
"... is neither valid JSON nor valid YAML ..." or "This is not a WSDL 1.1 document ..."	Use an OpenAPI 3 or Swagger 2 document, or a WSDL 1.1.
"Set the option <code>read_operations</code> : the operations that only read data ..."	List the SOAP operations that only read, separated by commas.
"Set the option <code>auth</code> : <code>oauth</code> (each person signs in with their own account, recommended), <code>api_key</code> , <code>bearer</code> or <code>basic</code> ..., or <code>none</code> ."	Choose how people sign in.
"With <code>auth</code> ..., set the option <code>secret</code> ..." or "With <code>auth</code> <code>basic</code> , set the option <code>username</code> ..."	Fill in the shared key, token or password (and the user name for <code>basic</code>).
"With <code>auth</code> <code>oauth</code> , set the options <code>authorize_url</code> and <code>token_url</code> ..."	Copy both addresses from the system's OAuth documentation, or use an OpenAPI document that lists them.
"... must start with <code>https://</code>"	Use the <code>https://</code> address.
"... names a cloud metadata or internal host, which SourceLace never calls."	Use the API's public address.
"...: ... leads to a private or reserved network address, so SourceLace did not call it."	SourceLace's cloud only calls addresses on the internet. Use a public address, or a self-hosted SourceLace inside your network.
"The address of ... could not be found. Check its spelling."	Fix the host name in <code>base_url</code> , <code>endpoint</code> , or the document.
"... sign-in to ... failed (...). Connect the source again."	The system's own reason is in brackets. Check that the OAuth client's redirect URL is exactly <code>https://sourcelace.onrender.com/connect/callback</code> , and the client ID and secret.
"... has no operation Use <code>search_schema</code> to list them."	Use an operation name that <code>search_schema</code> shows.
"... is a POST, which changes data, so query cannot run it. ..."	Changes go through <code>propose_change</code> , and only for operations an admin switched on.

What you see	What to do
"... is not switched on as a change for An admin can switch it on."	A SourceLace admin turns on Allow changes and names the operation.
"... needs Use describe_object to see its parameters." or "... has no parameter"	Fix the query's parameters.
"... has had ... requests in the last minute, the most SourceLace sends it. ..."	Wait the time shown, or ask for more rows per request.
"... did not answer within ... seconds. ..." or "... sent an answer larger than 10 MB. ..."	Ask for fewer rows or fields.
"... answered with ... instead of JSON, so SourceLace could not read it. ..."	Check base_url and that the operation returns JSON.
"... uses SOAP encoding (use=encoded), which SourceLace does not support."	That operation cannot be used.
"SOAP sources are read-only for now: turn changes off for this source."	Turn Allow changes off.