

Python SDK for izibuddy

Python SDK for izibuddy

InZeMobile

TABLE DES MATIÈRES

Resources	5
Partie 1 — Getting started	
Chapter I — What this SDK is.....	9
Chapter II — Install and first call.....	11
Chapter III — Authentication and sessions.....	13
Partie 2 — Using the client	
Chapter IV — The shape of the client	17
Chapter V — Errors	19
Chapter VI — Working with templates	21
Chapter VII — Bytes, pages and slashes.....	23
Partie 3 — Keeping it true	
Chapter VIII — Regenerating after an API change.....	27
Chapter IX — Publishing and consuming	29
Appendices	
Appendix A — The twenty-six resources.....	33

Resources

The SDK lives in `src/sdk/python/` of the IziBuddy repository :

- `izibuddy/client.py` — transport, session, typed errors, pagination. Written by hand ; the generator never touches it.
- `izibuddy/_generated.py` — the 252 operations and 94 models, produced from the backend's OpenAPI schema.
- `izibuddy/cli.py` — the `izibuddy` command, documented in book 13.
- `pyproject.toml` — no runtime dependencies ; Python 3.9 or later.

The generator is `src/sdk/generate.py` . It boots the backend in-process, reads its schema, and rewrites the generated file of this SDK and of its TypeScript twin :

```
python3 src/sdk/generate.py
```

The guard is `tests/test_sdk_drift.js` . It regenerates into a throwaway directory and compares byte for byte, naming the first line that differs. It never rewrites anything, and it skips rather than passing when the backend's dependencies are absent.

The backend it talks to is `src/api/` , one module per domain, documented in book 09. Method names here are those modules' function names.

Télécharger

- Format PDF (11-SDK-python_v1.3_en_20261008.pdf)
- Format EPUB (11-SDK-python_v1.3_en_20261008.epub)
- Format Markdown (11-SDK-python_v1.3_en_20261008.md)

Partie 1

Getting started

Chapter I — What this SDK is

The IziBuddy backend is a FastAPI application exposing **252 operations** across **26 resource groups** — templates, tags, groups, users, versioning, scoring, git, analytics, licensing, and the rest. This SDK is a Python client for all of them.

1.1 — Generated where it should be, written where it matters

Two hundred and fifty-two methods are not written by hand. They are **generated from the backend’s own OpenAPI schema**, which FastAPI derives from the route decorators and the Pydantic models. The generator lives at `src/sdk/generate.py` and produces `izibuddy/_generated.py`.

Everything else — the transport, the session, the typed errors, pagination — is **written by hand** in `izibuddy/client.py`, and the generator never touches it. The split is the design :

- an endpoint added to `src/api/` appears in the SDK by regenerating, not by remembering ;
- a decision about how errors should feel is made once and stays made.

The alternative was considered and rejected. A hand-written client over 252 operations drifts at the first route change, and a drifting client does not announce itself : it imports, it runs, and it is wrong. This repository has already paid that price once, in a different layer — six fields silently lost between the POO²L engine and the wrapper that was supposed to pass them through.

1.2 — What keeps it honest

`tests/test_sdk_drift.js` regenerates both SDKs into a throwaway directory and compares byte for byte with what is committed. If the backend moved and the SDK did not, that test is red and names the first line that differs. It never rewrites anything : the remedy is one command, printed in the failure.

1.3 — What this SDK is not

It is not a re-implementation of the backend’s rules. It does not validate prompt syntax, resolve includes, or know what a POO²L section is — the engine does that, server-side, and the SDK carries the request and the answer.

It is not an abstraction over the HTTP API either. Method names are the names of the Python functions in `src/api/`, so `izi.templates.render_template_by_id` is `render_template_by_id` in `src/api/templates.py`. You can grep from one to the other. That is deliberate : an SDK whose vocabulary differs from its server’s forces every reader to hold two maps at once.

Skills and publication

Generated models accept `document_type="skill"` for create/update and preserve UUIDs on conversion. Regenerate the SDK from the backend schema after updating. `client.templates.publish_document(body={...})` exposes the compare-and-publish request (`id`, `UUID token`, `base_revision`, `document`). Keep private drafts in client storage; do not call create/update on every draft save. HTTP 409 is a conflict to review, not permission to overwrite. Keep a pending draft until `publication_pending` is false. Do not automatically publish its dependencies.

Use the render endpoint with `target="skill"`; the emission contains SKILL.md text and relative files encoded as base64. The caller chooses a destination, validates paths and writes all files. Source publication and installed output are separate lifecycles.

Templates and document nature

A **template** is the common reusable entity. Its **Nature** is `prompt`, `skill` or a custom identifier such as `mod` or `acme.review`. Create **New prompt** or **New skill**, then edit Nature in Properties when needed. Changing nature preserves the UUID, references, parameters and history. The leaf template owns its type; includes never change it. Mixed inheritance can be allowed with warnings or denied.

```
@@@template review_assistant
@@@type skill
@@@description Reusable review instructions
@@@section rules
Separate observations from assumptions.
```

Use `type:acme.review` to search a custom nature. The **Example** flag replaces the old Template flag; `flag:example` finds these starting examples. `flag:draft` continues to find private drafts. A prompt copies the rendered text, a skill exports SKILL.md and resources, and other types export composed text without claiming a specialized package format.

Chapter II — Install and first call

2.1 — Install

The package lives in the repository, at `src/sdk/python/`, and has **no dependencies** — it uses `urllib` from the standard library. That is a deliberate constraint: a client whose job is to make HTTP calls should not force a dependency tree on the program that embeds it.

```
pip install -e src/sdk/python          # depuis une copie du dépôt
```

Is it on PyPI? Not yet, and this book will say so until it is. The distribution is named `izibuddy` — the same word as the command — it is built and verified on every release run, and publishing it is one tag away:

```
git tag sdk-v0.1.0 && git push origin sdk-v0.1.0
```

`.github/workflows/sdk-release.yml` builds the wheel and the source distribution, installs the wheel into a clean virtual environment **outside the repository** and runs the CLI from it, then publishes to PyPI through trusted publishing — no token stored anywhere. Until that tag is pushed, `pip install izibuddy` fetches nothing, and claiming otherwise would be disproved in ten seconds.

Without any checkout, and with access to the repository:

```
pip install "izibuddy @ git+https://github.com/Jacques66/appli-izibuddy@main#subdirectory=src/"
```

Or, without installing, put it on the path:

```
import sys
sys.path.insert(0, "src/sdk/python")
```

Python 3.9 or later.

2.2 — The first call

```
from izibuddy import IziBuddy

izi = IziBuddy("http://127.0.0.1:8765")
print(izi.api.health())
```

```
{'status': 'healthy',  
  'components': {'config_manager': True, 'templates_manager': True, ...},  
  'engine': {'available': True, 'capable': True, ...}}
```

`health` needs no authentication, which makes it the right first call: it proves the base URL, the port and the network path in one line, before any question of credentials.

2.3 — Finding the port

The backend reads `[server] port` from `izibuddy.ini`, but a launch may override it. When in doubt the running process says so on startup:

```
INFO:      Uvicorn running on http://127.0.0.1:8765
```

2.4 — What the client is

`IziBuddy` holds three things and nothing else: the base URL, the token when there is one, and a timeout. It is cheap to construct and safe to keep for the life of a program.

```
izi = IziBuddy("http://127.0.0.1:8765", timeout=60.0)  
izi                                     # <IziBuddy http://127.0.0.1:8765 (anonyme)>
```

The 26 resources are attached at construction, so `izi.templates`, `izi.tags_v2` and the rest exist immediately — no lazy loading, nothing to await.

Chapter III — Authentication and sessions

3.1 — The model

The backend authenticates with a **JWT bearer token**. `POST /api/auth/login` returns one; every protected route expects it in an `Authorization: Bearer ...` header. There is no server-side session to invalidate — which is why logging out is a client-side act, and nothing more.

3.2 — Logging in

```
izi = IziBuddy("http://127.0.0.1:8765")
izi.login("alice", "secret")
izi.authentication.get_me()
```

`login` is one of the few hand-written methods on the client rather than a generated one, and for a reason: it **remembers** the token. The generated `izi.authentication.login(...)` would return the same payload and leave you to carry the token yourself.

```
izi.token          # 'eyJhbGciOiJIUzI1NiIs...'
izi.logout()       # oublie le jeton, rien de plus
```

3.3 — Carrying a token you already have

A service that receives a token from elsewhere — a web session, a job runner — passes it straight in:

```
izi = IziBuddy("http://127.0.0.1:8765", token=jeton_recu)
```

3.4 — Knowing before you ask

Two calls need no token and answer the questions worth asking first:

```
izi.authentication.auth_status()
# {'auth_required': True, 'has_users': False, 'show_register': True}

izi.authentication.get_registration_config()
# {'success': True, 'self_registration_enabled': True,
#   'has_domain_restriction': False, 'email_enabled': False}
```

`has_users: False` on a fresh instance means the first account still has to be created — `izi.authentication.register(...)`. `has_domain_restriction` says whether [registration] `allowed_domains` will refuse an address.

3.5 — When the token is refused

An expired or missing token raises `AuthError`, which is also what a permission failure raises — 401 and 403 are the same problem from the caller's side: *this identity may not do this*.

```
from izibuddy import AuthError

try:
    izi.admin.restart_application()
except AuthError as e:
    print(e.statut, e.detail)      # 403 'Admin privileges required'
```

Partie 2

Using the client

Chapter IV — The shape of the client

4.1 — Resources are the backend's tags

Every FastAPI route carries a tag, and the SDK groups its methods by that tag. The mapping is mechanical, so there is nothing to memorise :

backend tag	attribute
templates	izi.templates
authentication	izi.authentication
tags-v2	izi.tags_v2
admin-alerts	izi.admin_alerts
template-builder-config	izi.template_builder_config

Twenty-six of them. The full list is in the appendix, with the operation count of each.

4.2 — Method names are the backend's function names

FastAPI builds an operation id from the handler's name and its path. The generator strips the path part back off, so what remains is the name the backend developer chose :

```
izi.templates.render_template_by_id("abc123")    # def render_template_by_id(...)
izi.tags_v2.merge_tags("a", "b")                # def merge_tags(...)
```

Two routes in the same group that share a handler name — the same function serving `GET` and `POST` — are disambiguated by appending the verb, and only then.

4.3 — Arguments follow the route

Path parameters come first, positionally, in the order the URL states them. A request body is next, as `body`. Query parameters are keyword-only, so they can never be passed by accident :

```
izi.templates.render_template_by_id("abc123", delivery="split")
#                               ^ chemin           ^ requête, nommée
izi.templates.update_template("abc123", {"content": nouveau_texte})
#                               ^ chemin  ^ corps
```

A query parameter left at `None` is **not sent**. That matters: `?delivery=None` is not the same request as `?delivery=` and neither is the same as sending nothing, and the backend distinguishes them.

4.4 — What comes back

A JSON response is decoded and returned as-is — a `dict`, a `list`, whatever the endpoint produces. Where the backend declares a Pydantic model, the return is typed with a `TypedDict` of the same shape, generated alongside; where it returns a bare `dict`, the type is `Any`, which is the honest answer.

Ninety-four models are generated. They are type hints, not validators: nothing is checked at runtime, because the backend already checked, and checking twice means two answers to keep in agreement.

4.5 — Non-JSON responses

Several endpoints do not return JSON — user-guide pages are markdown, analytics exports are CSV, attachments are bytes. Those come back as `bytes`, not as a failed JSON parse:

```
guide = izi.documentation.get_guide_page("syntax")    # bytes (markdown)
csv   = izi.analytics.export_analytics_csv()         # bytes
```

Chapter V — Errors

5.1 — One family, six names

Every refusal raises a subclass of `IziBuddyError`, and each one carries the same four facts : the status, the decoded detail, the method and the path.

exception	status	means
<code>AuthError</code>	401, 403	no token, expired token, or not allowed
<code>NotFound</code>	404	
<code>Conflict</code>	409	the state forbids it — a duplicate id, a locked record
<code>ValidationError</code>	422	the body does not satisfy the backend's model
<code>ServerError</code>	5xx	
<code>IziBuddyError</code>	anything else, and 0 for a network failure	

```
from izibuddy import IziBuddyError, NotFound, ValidationError

try:
    izi.templates.get_template("nexistepas")
except NotFound as e:
    print(e.statut, e.methode, e.chemin)      # 404 GET /api/templates/get/nexistepas
except IziBuddyError as e:
    ...
```

5.2 — 422 is the useful one

FastAPI does not answer a malformed body with a sentence. It answers with a list saying which field, why, and what was received — and `detail` carries that list untouched :

```
try:
    izi.authentication.register({"username": "x", "password": "y",
                                "email": "sdk@demo.local"})
except ValidationError as e:
    for f in e.detail:
        print(f["loc"], "→", f["msg"])
```

```
['body', 'email'] → value is not a valid email address: The part after the
@-sign is a special-use or reserved name that cannot be used with email.
```

A client that flattened that into a string would throw away the only part a caller can act on: `loc`.

5.3 — A network failure is not a status

Unreachable host, refused connection, timeout: `IziBuddyError` with `statut == 0` and the reason in `detail`. It is deliberately the same family — a caller that wants to retry, log or give up treats them together — but the zero says plainly that the server never answered, which is a different fact from a server that answered badly.

5.4 — Not every failure is an exception

This is the caveat to read twice. A number of endpoints report failure in the body, with HTTP 200:

```
izi.templates.get_template_by_id("nexistepas")
# {'success': False, 'error': 'Template nexistepas not found'}

izi.templates.render_template_by_id("nexistepas")
# {'success': False, 'error': "Template with id 'nexistepas' not found"}
```

No `NotFound` is raised, because nothing went wrong at the HTTP level — the server understood the question and answered it. A caller that only catches exceptions will read `result["prompt"]`, get a `KeyError`, and blame the SDK.

The rule, until the backend is uniform: **catch the exception AND check success when the payload has one.**

```
r = izi.templates.render_template_by_id(uid)
if not r.get("success", True):
    raise RuntimeError(r.get("error", "échec sans message"))
```

The SDK does not translate `{'success': False}` into an exception on your behalf. It could — and it would be wrong the day an endpoint returns that key with a meaning of its own. Passing the body through unchanged is the only answer that stays true.

5.5 — What the SDK never does

It does not retry. It does not swallow. It does not translate a failure into a `None` that a caller might mistake for an empty result. Retry policy belongs to the program that knows whether the operation is idempotent — and of the 252 operations, many are not.

Chapter VI — Working with templates

Templates are the largest resource — **57 operations**, more than a fifth of the whole API. This chapter walks the round trip that matters, with the output of an actual run.

6.1 — Create

A prompt's `content` is POO²L source. The backend stores it and the engine renders it; the SDK carries the string and does not read it.

```
source = (
    "###template sdk_note\n"
    "###name Note du SDK\n"
    "###section corps : Corps\n"
    "Bonjour depuis le SDK.\n"
)
izi.templates.create_template({
    "name": "Note du SDK",
    "template_id": "sdk_note",
    "content": source,
})
# {'success': True, 'message': 'Template created successfully', 'version': '0.1'}
```

Note what the answer does **not** contain: the new record's uid. Creation reports the outcome, not the object.

6.2 — Two identifiers, and they are not interchangeable

```
p = izi.templates.list_all_templates()["templates"][0]
p["id"]          # '9e0fe140-239a-4e65-a57d-db8b2f402212' - storage identity
p["template_id"] # 'SDK_NOTE'                          - POO2L identity
```

`id` is the uid the storage layer assigns; `template_id` is the name POO²L resolves — case- and diacritic-insensitive, which is why it comes back upper-cased. Most routes want the **uid**. Passing a `template_id` where a uid is expected fails with a 404 that reads oddly, because the backend looked up exactly what it was given.

6.3 — Render

```
izi.templates.render_template_by_id(p["id"])
```

```
{'success': True,
 'rendered': '# Note du SDK (SDK_NOTE)\n\n## Corps\n\nBonjour depuis le SDK.\n',
```

```
'template_id': 'SDK_NOTE',  
'warnings': [{ 'type': 'missing_description', 'message': "MISSING_DESCRIPTION: ..." } ] }
```

The `#` and `##` are POO²L structure headings, emitted by the engine — the level-1 line names the instantiated prompt, the level-2 line its section. They are part of the contract, not decoration.

`warnings` is where advisory diagnostics land. They never block a render, and a client that ignores the key will never know it discarded them.

6.4 — Delivery : where the attachments go

Attachments live in the record, never in the prompt text. `delivery` decides what the render returns :

```
izi.templates.render_template_by_id(uid, delivery="attach") # défaut : texte propre  
izi.templates.render_template_by_id(uid, delivery="inline") # texte + bloc de pièces  
izi.templates.render_template_by_id(uid, delivery="split") # texte + bloc, séparés
```

The assembly is done **server-side** in all three cases, so an application and an extension asking for the same delivery get byte-identical text.

6.5 — Render with parameters

A prompt declaring `@@@param` is instantiated by the `POST` form, which takes the values :

```
izi.templates.render_template_with_params(uid, {"parameters": {"lang": "fr"}})
```

6.6 — Delete, and what delete means

```
izi.templates.delete_template(uid)  
# { 'success': True, 'message': 'Template deleted successfully', 'warnings': [] }
```

That is a move to the trash, not an erasure — `izi.templates.list_trash()` still lists it and `restore_template` brings it back. The irreversible form is `purge_template`, and it is a different method for that reason.

Chapter VII — Bytes, pages and slashes

Three places where a generated client would be wrong if nobody had thought about them.

7.1 — Path parameters that accept slashes

Seven routes declare a parameter with FastAPI's `:path` converter — one on collections, six on tags :

```
/api/templates/collections/{path:path}
/api/tags/{tag_name:path}
/api/tags/color/{tag_name:path}          (et acronym, icon, short_name, show_icon)
```

Those parameters are **allowed to contain** `/` : a nested collection is `Marketing/2026`, a hierarchical tag is `client/acme/urgent`. Percent-encoding them would change the route being called.

The trap is that the OpenAPI schema **erases the converter**: `/api/tags/{tag_name:path}` appears there as `/api/tags/{tag_name}`, indistinguishable from an ordinary segment. A generator reading the schema alone cannot tell the difference — so this one reads the converters out of `src/api/` instead, and encodes accordingly :

```
izi.templates.update_collection("Marketing/2026", {...})  # les `/` survivent
izi.templates.get_template_by_id("a/b")                  # encodé : un seul segment
```

The list is exported as `izibuddy.CHEMINS_LIBRES` if a caller needs to reason about it.

7.2 — Responses that are not JSON

```
guide = izi.documentation.get_guide_page("syntax")      # bytes – markdown
csv   = izi.analytics.export_analytics_csv()            # bytes – CSV
octets = izi.templates.get_template_attachment(uid, "nda.pdf") # bytes
```

The client looks at `Content-Type` and returns raw `bytes` when it is not JSON, rather than failing to parse something that was never JSON. Attachment bytes in particular are the point of that endpoint : it is the only place in the system that decodes them.

7.3 — Caching reads

A loop that renders a hundred templates re-reads the same list, the same settings, the same tags. Off by default ; one argument turns it on :

```

izi = IziBuddy(url, token=t, cache_ttl=60)          # secondes
izi.templates.list_all_templates()                  # 80.7 ms
izi.templates.list_all_templates()                  # 0.014 ms
izi.cache_stats()      # {'entries': 1, 'hits': 1, 'misses': 1, 'ttl': 60}

```

Three rules, and they are not options.

`GET` only. Caching a write would return a response without writing, so the VERB decides — not the caller.

The key carries the token. Two users do not see the same templates, and a key blind to identity would serve one what the other read. That is a leak, not an optimisation.

Errors are never kept. A one-second outage must not last as long as the TTL.

It is off by default, and that is deliberate: a cache nobody asked for serves yesterday's answer to someone who believes they are reading today.

7.4 — What a cache cannot guess

```

izi.templates.update_template(uid, {...})
izi.cache_clear()      # sinon la lecture suivante peut être l'ancienne

```

The client does not know which reads a write invalidates — a rename changes a listing, a share changes what another user sees — and pretending to know would be worse than doing nothing. Clear it, or use a TTL short enough that the staleness is one you accept.

`stale_on_error=True` goes the other way: when the backend stops answering, an expired value is served rather than raising. Off by default, because serving stale silently makes an outage look like normal operation.

7.5 — Long lists

Several list endpoints take `limit / offset`. `pages()` walks them without holding everything in memory:

```

for p in izi.pages("GET", "/api/templates/list", cle="templates", taille=200):
    print(p["template_id"])

```

`cle` is required because the key holding the list differs between endpoints — `templates` here, something else elsewhere. Guessing it would be the kind of convenience that works until it silently returns nothing.

Partie 3

Keeping it true

Chapter VIII — Regenerating after an API change

8.1 — One command

```
python3 src/sdk/generate.py
```

It boots the backend application in-process, reads its OpenAPI schema, and rewrites two files :

```
src/sdk/python/izibuddy/_generated.py  
src/sdk/typescript/src/generated.ts
```

Nothing else is touched. `client.py` — transport, session, errors, pagination — is hand-written and stays where it is.

8.2 — The guard

```
node tests/test_sdk_drift.js
```

It regenerates into a throwaway directory and compares byte for byte with what is committed. Red means the backend moved and the SDK did not, and the failure names the first line that differs. It never rewrites anything : fixing it is the one command above, followed by a look at the diff.

Without the backend's dependencies installed the bench **skips** rather than passing — a check that cannot run must not report success.

8.3 — What a diff will show you

A route added : a new method. A handler renamed : a method renamed — which is a breaking change for callers, and better seen in a diff than discovered by a user. A Pydantic model gaining a field : a `TypedDict` gaining a key.

8.4 — A trap worth knowing

Two handlers registered on the same method and path are legal Python and a silent problem. FastAPI serves the **first** and the schema keeps **one** entry — so the generated method may carry the name of the handler that never runs.

That case exists today: `GET /api/templates/sharing/{template_id}` is declared twice in `src/api/templates.py`, by `get_template_sharing` and by `get_template_sharing_info`. The SDK exposes `get_template_sharing_info` ; the backend

answers with `get_template_sharing`. The two return compatible shapes, so nothing breaks — but the name lies, and no amount of regeneration will fix it. The fix is in `src/api/`.

Chapter IX — Publishing and consuming

Two audiences share this chapter. A **consumer** wants one install line. A **maintainer** wants to know what stands between the repository and that line.

9.1 — Where it stands today

Neither registry carries the packages yet. That sentence is in this book on purpose: a claim a reader disproves in ten seconds costs more than the gap it admits, and `pip install izibuddy` is a ten-second test.

What exists is everything up to the act of publishing. The distribution is named `izibuddy` — the same word as the command, the same word this book uses throughout — its metadata is complete, and every release run **installs the built artifact into a clean environment outside the repository** and runs the CLI from it before anything is published.

That last check is the one that matters. `pip install -e .` succeeds on a package that is missing files, because the source tree is right there; an install from a wheel does not.

9.2 — Installing today

From a checkout:

```
pip install -e src/sdk/python
```

Without a checkout, with access to the repository:

```
pip install "izibuddy @ git+https://github.com/Jacques66/appli-izibuddy@main#subdirectory=src/"
```

Pin a tag rather than `main` for anything that has to reproduce:
`...@sdk-v0.1.0#subdirectory=...`

9.3 — Installing once published

```
pip install izibuddy  
pip install "izibuddy==0.1.0"      # épinglé
```

The console script comes with it, so `izibuddy --version` answers immediately. Nothing else is pulled in: the package declares **no runtime dependencies**.

9.4 — Publishing, for a maintainer

One tag:

```
git tag sdk-v0.1.0 && git push origin sdk-v0.1.0
```

`.github/workflows/sdk-release.yml` runs the drift and CLI benches first — publishing an SDK that has fallen behind its backend is worse than publishing nothing, since nobody would know that what they installed lies — then builds both packages, verifies the standalone install, and publishes.

PyPI uses **trusted publishing**: the project is configured once to accept that workflow over OIDC, and no token is stored in the repository. npm needs an automation token, held by a `release` environment where a required reviewer belongs if you want one — publishing is irreversible, and a version number is never reusable.

The full checklist, including the one-time settings and the public-versus- private decision on the npm scope, is `src/sdk/RELEASE.md`.

9.5 — Rehearsing

Actions → *SDK release* → *Run workflow*, `dry-run` checked. Everything runs, the artifacts are downloadable from the run, and both publish jobs are skipped: they are guarded on the tag, and a manual run carries none.

Appendices

Appendix A — The twenty-six resources

Every attribute of the client, with the backend tag it comes from and the number of operations it carries. Counts are what the schema declares.

attribute	backend tag	operations
izi.admin	admin	8
izi.admin_alerts	admin-alerts	7
izi.analytics	analytics	12
izi.api	<i>(untagged — root routes)</i>	5
izi.authentication	authentication	17
izi.config	config	5
izi.diagnostic	diagnostic	3
izi.diff	diff	1
izi.documentation	documentation	6
izi.events	events	4
izi.git	git	12
izi.groups	groups	19
izi.invitations	invitations	9
izi.license	license	5
izi.migration	migration	7
izi.presets	presets	4
izi.template_builder_config	template-builder-config	14
izi.template_structure	template-structure	4
izi.templates	templates	57
izi.scoring	scoring	5
izi.status	status	4
izi.tags	tags	11

attribute	backend tag	operations
izi.tags_v2	tags-v2	10
izi.users	users	16
izi.versioning	versioning	5
izi.webhooks	webhooks	2

252 operations. The listing is generated from the same schema as the code, so it cannot disagree with it — and `tests/test_sdk_drift.js` fails if it does.

To enumerate them at runtime :

```
import izibuddy
sorted(izibuddy.RESSOURCES)
[m for m in dir(izi.templates) if not m.startswith("_")]
```


