

Documentación » Bridge Python » Uso del Bridge Python

Uso del Bridge Python

Uso completo del Bridge Python

Anonymous · 08/10/2026

Uso del Bridge Python

Repasa cómo apuntar `Dispatcher` a tu instalación del Dispatcher, despachar operaciones y manejar los errores que puedan ocurrir.

Ubicar el Dispatcher

Dispatcher no sabe dónde vive el código PHP en disco. Eso se resuelve vía la variable de entorno `BACKBONE_DISPATCHER_AUTOLOAD` (apuntando al `vendor/autoload.php` de tu instalación del [Dispatcher](#)):

```
autoload=/ruta/al/dispatcher/vendor/autoload.php
export BACKBONE_DISPATCHER_AUTOLOAD="$autoload"
```

```
from libredte_lib_core_bridge import Dispatcher

dispatcher = Dispatcher()
```

O pasando `autoload_path` explícito al construirlo, sin necesidad de la variable de entorno:

```
autoload_path = '/ruta/al/dispatcher/vendor/autoload.php'

# El entorno puede ser 'prod' (default), 'dev' o 'test'.
dispatcher = Dispatcher(
    environment='prod',
    debug=False,
    autoload_path=autoload_path,
)
```

Despachar operaciones

Con el dispatcher ya listo, cada llamada indica la operación y sus parámetros como argumentos con nombre:

```
result = dispatcher.dispatch(  
    'billing.identifier.caf_loader::load',  
    xml=xml_base64,  
)
```

El identificador de la operación sigue la forma "package.component.worker:operation", y los parámetros se pasan como argumentos con nombre. Revisa [Funcionalidades](#) de la biblioteca PHP para ver qué paquetes, componentes y workers (y por lo tanto qué operaciones) están disponibles.

Manejo de errores

`dispatch()` nunca deja pasar un error de PHP sin tipar: lanza una excepción Python (`derafu_backbone_bridge.BackboneBridgeError` o una de sus subclases, como `LibredteLibCoreError` para errores propios de `libredte-lib-core`) en vez de una excepción PHP cruda.

Última actualización el 08/10/2026