⚠️ Errors

Contents

- Base Exception

Exceptions are important part of the `pywa` library. They are used to tell you what went wrong and why.

Most of the exceptions are raised when you try to do something like sending a message:

```python
import logging
from pywa import WhatsApp
from pywa.types import Button
from pywa.errors import InvalidParameter

wa = WhatsApp(...)

try:
    wa.send_message(..., buttons=[
        Button(title="click 1", callback_data="click"),
        Button(title="click 2", callback_data="click"),
    ])
except InvalidParameter as e:  # duplicate callback_data in buttons (`click`)
    logging.error(f"Duplicated callback_data in buttons: {e}")
```

But there are also errors that are not raised, but can be returned in a message status.

For example, you can sometimes try to `send_message()` to a user that the last time you sent a message to him was less than 24 hours ago, if this message is not a `Template` message, you will not get an exception, but you will get `MessageStatus` on `FAILED` status with .error attribute with value of `ReEngagementMessage`.

The same goes for media messages: if you try to send a invalid media (unsupported file type, too big file, invalid url, etc.), You will not get the exception, when you try to send the message, but in the message status error attribute (`MediaUploadError`).

That's why it's important to always register a handler for failed status messages, so you can know when a message failed to send:

```python
import logging
from pywa import WhatsApp
from pywa.types import MessageStatus
from pywa import filters as fil

wa = WhatsApp(...)

@wa.on_message_status(fil.message_status.failed)  # filter for failed message statu
def handle_failed_message(client: WhatsApp, msg: MessageStatus):
    logging.error("Message failed to sent to %s: %s. details: %s",
        status.from_user.wa_id, status.error.message, status.error.details
    )
```

You can also handle specific errors, for example, if you want to handle only media errors:

```python
import logging
from pywa import WhatsApp
from pywa.errors import MediaUploadError, MediaDownloadError, ReEngagementMessage
from pywa.types import MessageStatus
from pywa import filters as fil

wa = WhatsApp(...)

wa.send_message(to="972501234567", text="Hello")  # this conversation window is clo

wa.send_image(  # this image does not exist
    to="972501234567",
    image="https://example.com/this-image-does-not-exist.jpg",
    caption="Not found"
)

wa.send_document(  # this document is too big
    to="972501234567",
    document="https://example.com/document-size-is-too-big.pdf",
    filename="big.pdf"
)

@wa.on_message_status(fil.message_status.failed_with(ReEngagementMessage))
def handle_failed_reengagement(client: WhatsApp, msg: MessageStatus):
    logging.error("Message failed to sent to %s: %s. details: %s",
        status.from_user.wa_id, status.error.message, status.error.details
    )

@wa.on_message_status(fil.message_status.failed_with(MediaUploadError, MediaDownloa
def handle_failed_sent_media(client: WhatsApp, msg: MessageStatus):
    if isinstance(msg.error, MediaUploadError):
        msg.reply_text(f"Sorry, I can't upload this file: {msg.error.details}")
    elif isinstance(msg.error, MediaDownloadError):
        msg.reply_text(f"Sorry, I can't download this file: {msg.error.details}")
```

Another example for “incoming” errors is for unsupported messages: if the user sends unsupported message type (like pool), you will get the message with type of `UNSUPPORTED`

and with error of `UnsupportedMessageType`.

```python
from pywa import WhatsApp
from pywa.types import Message
from pywa import filters as fil

wa = WhatsApp(...)

@wa.on_message(fil.unsupported)
def handle_unsupported_message(client: WhatsApp, msg: Message):
    msg.reply_text("Sorry, I don't support this message type yet")
```

All the exceptions are inherited from `WhatsAppError`, so you can catch all of them with one exception:

```python
from pywa import WhatsApp
from pywa.errors import WhatsAppError

wa = WhatsApp(...)

try:
    wa.send_message(...)
except WhatsAppError as e:
    print(f"Error: {e}")
```

Base Exception

class **pywa.errors.WhatsAppError**

Bases: `Exception`

Base exception for all WhatsApp errors.

- 'Cloud API Error Codes' on developers.facebook.com.
- 'Flow Error Codes' on developers.facebook.com.

Variables:

- **error_code** – The error code.
- **error_subcode** – The error subcode (optional).
- **type** – The error type (optional).
- **message** – The error message.
- **details** – The error details (optional).
- **fbtrace_id** – The Facebook trace ID (optional).

- **href** – The href to the documentation (optional).
- **raw_response** – The `httpx.Response` obj that returned the error (optional, only if the error was raised from an API call).

The exceptions are divided into 5 categories:

Sending Messages Errors

- `SendMessageError`
- `MessageUndeliverable`
- `ReEngagementMessage`
- `UnsupportedMessageType`
- `RecipientNotInAllowedList`
- `InvalidParameter`
- `MissingRequiredParameter`
- `MediaDownloadError`
- `MediaUploadError`
- `TemplateParamCountMismatch`
- `TemplateParamFormatMismatch`
- `TemplateNotExists`
- `TemplateTextTooLong`
- `GenericError`
- `UnknownError`
- `AccessDenied`
- `ServiceUnavailable`
- `RecipientCannotBeSender`
- `BusinessPaymentIssue`
- `IncorrectCertificate`
- `AccountInMaintenanceMode`

Flows Errors

- `FlowError`
- `FlowBlockedByIntegrity`

- `FlowUpdatingError`
- `FlowPublishingError`
- `FlowDeprecatingError`
- `FlowDeletingError`

Authorization Errors

- `AuthorizationError`
- `AuthException`
- `APIMethod`
- `PermissionDenied`
- `ExpiredAccessToken`
- `APIPermission`

Rate Limit Errors

- `ThrottlingError`
- `ToManyAPICalls`
- `RateLimitIssues`
- `RateLimitHit`
- `SpamRateLimitHit`
- `ToManyMessages`

Integrity Errors

- `IntegrityError`
- `TemporarilyBlocked`
- `AccountLocked`