汎用ログ出力 Webhook — Webhook 仕様書

Denwaban の AI 受付が通話を終えたときに、お客様がご指定の URL へ通話ログを JSON で送信します。本書は、その Webhook を受け取る側を実装するための仕様書です。

項目	値
仕様バージョン	`0.1.0` (**確定版**)
最終更新	2026-08-14
提供元	Global Internet Japan Inc.

> **確定版です。** 実装が完了し、**本書の全項目を実機で突き合わせ済み**です (実際の着信で通話ログの配信・録音の取得・認証の失敗時の応答まで確認しました)。 ドラフト時点からの差分は変更履歴にまとめてあります。
>
> ドラフトからの主な変更は次の 2 点です。**どちらも受信側の実装で必要な作りは変わりません。**
>
> - **要約の待ち時間の上限 (30 秒) を明記**しました (2 章)。現時点の 実装値で、運用状況を見て変更することがあります (変更時は改訂版をお渡しします)
> - **送信は録音の準備完了を待ちません** (2 章 / 6 章)。 録音が有効な通話には常にリンクを付けて送信し、**まだ準備できていない場合は取得時に** `404` `{"status":"pending"}` が返ります (時間を置いて再試行してください)。 `404` `{"status":"pending"}` の再試行は元から必要でした

1. 概要

- **通話が終了した時点**で、1 通話につき **1 回**送信します。**録音の準備完了は待ちません**
- 送信内容は JSON です。通話の基本情報・要約・文字起こし・**録音のダウンロードリンク**を含みます。**`recording.url` は `null` になることがあります。** 録音が無効な通話に加えて、 **録音が有効でもリンクを発行できなかった場合** (お客様が録音リンクのキーを未設定のときなど) も `null` です。**受信側は常に `recording.status` を見て分岐してください** (値の一覧は 6-3)。 リンクへのアクセス時にまだ録音が準備できていない場合は `404` `{"status":"pending"}` が返るため、 時間を置いて再試行してください (6-2)
- 録音の音声データそのものは送信しません。**リンク経由で取得**していただきます
- 送信先の URL と認証情報は、Denwaban の管理画面 (ツールの設定画面) で設定します

似た名前のツールとの違い

Denwaban には「**汎用 Webhook 連携**」という別のツールがあります。混同しないようご注意ください。

	汎用ログ出力 Webhook (本書)	汎用 Webhook 連携
向き	Denwaban → お客様 (送信)	Denwaban → お客様 (問い合わせ)

	汎用ログ出力 Webhook (本書)	汎用 Webhook 連携
タイミング	通話終了後	**通話中**
目的	通話ログの記録・連携	AI が外部 API を呼んで回答に使う

2. 送信タイミング

```
通話終了 → 要約の生成（最大 30 秒。失敗しても次へ進みます）
    ↓
Webhook を送信（1 通話 1 回）
    ↓
録音は準備でき次第、リンクから取得できるようになります
```

- **要約が生成できなかった場合も送信します。** その場合 `summary` は `null` になります。 要約が無いことを理由に送信が止まることはありません
- **要約は送信前に一度だけ待ちます。** 生成に時間がかかると、その分だけ送信も遅れます。 待ち時間の上限は **30 秒**です (現時点の実装値。運用状況を見て変更することがあります)。 上限を超えた場合は要約を打ち切り、 `summary` は `null` になります
- **後から要約が生成されても、追加の送信は行いません。** `summary` が `null` で届いた通話に 後から要約が付くことはないため、**受信側で待ち合わせる必要はありません**
- 録音は通話終了と同時には準備できません。**数秒〜数十秒の遅延**があります。 **送信はこれを待ちません。** 録音が有効な通話には `recording.status` が `available` の リンクを付けて送信し、**アクセスされた時点で準備できていなければ** `404` `{"status":"pending"}` を返します (時間を置いて再試行してください)
- 録音が無効な通話・録音が存在しない通話は `recording.status` が `unavailable` になり、 リンクは付きません。**録音の有無にかかわらず通話ログは送信します** (詳細は 6. 録音の取得)
- 通話が途中で切断された場合なども、 `call.status` に結果を入れて送信します

> **配信は保証されません。** 送信先が応答しない状態が続くと、7-3 の 再送を試みたうえで配信を打ち切ります。
>
> また、**ごくまれに通話ログ自体が送信されないことがあります。** 通話終了の情報が本ツールに 届かなかった場合で、この通話については再送も行われません (受信側からは、届かなかったのか そもそも通話が無かったのかを区別できません)。
>
> **取りこぼしを検知したい場合は、Denwaban の管理画面の通話履歴と受信済みデータを 突き合わせてください。**

3. HTTP 仕様

項目	値
メソッド	`POST`
Content-Type	`application/json; charset=utf-8`
プロトコル	**HTTPS のみ** (http:// は設定できません)
タイムアウト	10 秒
最大ボディサイズ	**1 MB**
User-Agent	`Denwaban-LogWebhook/<バージョン>`

> **受信側のボディサイズ上限は 1 MB 以上に設定してください。** 上限を下回ると、長い通話の 文字起こしを含む配信が `413` で拒否されます。 **`413` は 4xx なので再送されず、その通話ログは 失われます。**

1 MB を超える場合

長い通話では、文字起こしを含めた JSON が 1 MB を超えることがあります。その場合は **送信を取りやめるのではなく、1 MB に収まるように文字起こしを削って送信します。**

- 削るのは**文字起こしのみ**です。`call` / `summary` / `recording` は必ず送信します
- 削る順序は**古い発話から**です。通話の終盤 (結論が出ている部分) が残ります
- 削った場合は **`transcript.truncated` が `true`** になります

> **`transcript.truncated` が `true` になるのは、この「サイズ超過で削った」場合だけです。** 通話中の発話が配信の上限で届かなかった場合は、 **`truncated` は `false` のままです。** ツール側からは、発話がもともと無かったのか、 届かなかったのかを区別できないためです。

送信先 URL に設定できないもの

次の宛先は設定画面で登録できません。誤って社内ネットワークへ配信されることを防ぐためです。

- `http://` (HTTPS のみ)
- プライベート IP アドレス・ループバック・リンクローカルアドレス
- クラウドのメタデータサービス

ホスト名で登録した場合も、**送信のたびに名前解決の結果を検査**します。A / AAAA の **すべての結果**を検査し (IPv4 射影 IPv6 を含む)、**検査に通った IP アドレスへ接続を固定** します。検査の後にもう一度名前解決を行うと、その間に応答が差し替えられて内部ネットワークへ 接続してしまう恐れがあるためです。社内システムで受ける場合は、インターネットから到達できる 経路を用意してください。

リクエストヘッダ

ヘッダ	説明
`X-Denwaban-Event`	イベント種別。現在は `call.completed` のみ
`X-Denwaban-Delivery`	**配信 ID**。冪等キーとして使用してください (7 章)
`X-Denwaban-Timestamp`	**そのリクエストの送信時刻** (ISO 8601)。`Z` または明示的な UTC オフセットが必ず付きます。再送のたびに更新されます

期待するレスポンス

- **2xx を返してください**。それ以外は失敗とみなします
- レスポンスボディは参照しません。**ステータスコードを受け取った時点で配信は成功として確定**し、 本文の完了は待ちません。打ち切り方はプロトコルによって変わります。
  - **HTTP/1.1**: 本文を読み切らずに再利用することはできないため、**接続を閉じます**
  - **HTTP/2**: **該当のストリームだけを中断**し、接続は維持します (同じ接続を共有している他の配信には影響しません)

> **重要 — 2xx を返す前に受信内容を保存してください。 2xx を返した時点で配信は成功したとみなし、以後 Denwaban は再送しません。** 保存より先に 2xx を返すと、その後の処理に失敗した通話ログは**どちらにも残らず失われます**。とくに レスポンス後に処理が打ち切られる実行環境 (サーバーレス等) では確実に問題になります。

推奨する順序は次のとおりです。

1. **受信内容をそのまま保存する**。ここまでを同期で行う
2. **2xx を返す**
3. 重い処理 (録音の取得、業務システムへの登録など) は、**その後**に非同期で行う

1 の保存先は、プロセスが落ちても内容が残るものにしてください (データベースへのコミット、 永続キューへの登録など)。**プロセス内のメモリ上のキューや変数に入れただけでは不十分です。** 再起動やクラッシュで消え、送信側は再送しないため、通話ログが失われます。

保存に失敗した場合は 5xx を返してください。再送されます。処理は 10 秒でタイムアウトするため、 1 に時間のかかる処理を含めないでください。

---

4. 受信側の認証

送信先に認証をかけている場合、以下の 4 方式から選べます。設定画面で方式と値を指定してください。

方式	送信されるもの
`bearer`	`Authorization: Bearer <トークン>`
`basic`	`Authorization: Basic <base64(ユーザー名:パスワード)>` **※ユーザー名に `:` は使えません。** base64 に変換する前のバイト列は **UTF-8** です
`header`	任意のヘッダ名で任意の値 (例: `X-Api-Key: xxxxx`)
`query`	任意のクエリパラメータ名で任意の値 (例: `?api_key=xxxxx`)

認証情報は暗号化して保管し、設定画面には再表示しません。

`header` / `query` で指定できる名前には制限があります。**Denwaban が使用するヘッダ (`X-Denwaban-` で始まるもの)、および `Content-Type` / `Content-Length` などの HTTP の 基本的なヘッダは指定できません。** 配信そのものや重複排除の仕組みが壊れるためです。

`query` の場合、送信先 URL に既に含まれているクエリパラメータ名は指定できません。 同じ名前が 2 つ並ぶと、どちらの値が使われるかは受信側の実装によって異なり、**古い値で 認証されて `401` (再送されない) になり、配信が全滅する**恐れがあるためです。URL 側の パラメータを消すか、別の名前を使ってください。

> **`bearer` または `header` を推奨します。** `query` は認証情報が URL に含まれるため、 HTTPS で通信が保護されていても、**経路上のプロキシ・ロードバランサー・アクセスログ・監視 ツールに認証情報が平文で記録され続けます。** `query` を選ぶ場合は、受信側でアクセスログから 該当パラメータをマスクする設定を必ず行ってください。
>
> **将来の拡張予定**: リクエスト本文の HMAC 署名ヘッダの追加を検討しています。追加時も既存の 4 方式は維持し、署名の検証は任意とします。

---

5. ペイロード

5-1. 全体構造

```
{
  "specVersion": "0.1.0",
  "event": "call.completed",
  "connectionId": "conn_01J8XJZ9Q7WMBN4KTS2VD6HYFA",
  "deliveryId": "dlv_01J8XK2QW3E4R5T6Y7U8I9O0P1",
  "occurredAt": "2026-08-10T05:23:45.312Z",
  "call": {
    "id": "1496159ccda006c9323585ba8f360811",
    "status": "transferred",
    "startedAt": "2026-08-10T05:21:33.104Z",
    "endedAt": "2026-08-10T05:23:45.088Z",
    "durationSeconds": 132,
    "callerNumber": "819012345678",
```

```
    "calledNumber": "815012345678",
    "transferLabel": "サポート窓口",
    "pointsConsumed": 12
  },
  "summary": "配送日時の変更について。8月15日午前中への変更を希望。担当者から折り返し連絡することで合意。",
  "transcript": {
    "truncated": false,
    "text": "AI: お電話ありがとうございます。○○店でございます。\n発信者: 配送日を変更したいのですが。\nAI:
        かしこまりました。ご希望の日時をお伺いできますか。",
    "turns": [
      { "seq": 1, "speaker": "ai",     "text": "お電話ありがとうございます。○○店でございます。",
        "at": "2026-08-10T05:21:35.220Z" },
      { "seq": 4, "speaker": "caller", "text": "配送日を変更したいのですが。",
        "at": "2026-08-10T05:21:42.870Z" },
      { "seq": 9, "speaker": "ai",     "text": "かしこまりました。ご希望の日時をお伺いできますか。",
        "at": "2026-08-10T05:21:48.019Z" }
    ]
  },
  "recording": {
    "status": "available",
    "url": "https://denwaban-tool-log-webhook-prod-cx44nrax2q-an.a.run.app/recordings/
        QmFzZTY0dXJsLWVuY29kZWQtb3BhcXVlLXRva2Vu",
    "expiresAt": "2026-09-09T05:23:45.312Z",
    "contentType": "audio/mpeg"
  }
}
```


5-2. 項目一覧

トップレベル

キー	型	必須	説明
`specVersion`	string	✓	本仕様のバージョン。破壊的変更時に上がります
`event`	string	✓	現在は `call.completed` 固定
`connectionId`	string	✓	**接続 ID** (下記)
`deliveryId`	string	✓	配信 ID。**再送しても同じ値**です。冪等キーに使ってください
`occurredAt`	string	✓	**配信を作成した時刻** (ISO 8601、タイムゾーン付き)。**再送しても変わりません**。各リクエストの送信時刻は `X-Denwaban-Timestamp` を見てください
`call`	object	✓	通話の基本情報
`summary`	string | null	✓	AI が生成した通話の要約。**キーは必ず含まれます**。生成されなかった場合は値が `null`
`transcript`	object	✓	文字起こし
`recording`	object	✓	録音の情報

複数の設定から受け取る場合

1 つの受信サービスで、**複数のお客様や複数の設定から配信を受け取る**ことがあります。 その場合に配信元を区別するのが `connectionId` です。

- **設定 1 つにつき 1 つの値**が割り当てられ、その設定が続く限り変わりません
- 値そのものに意味はありません。**識別子としてのみ**扱ってください
- **データを保存する際は `connectionId` と組み合わせてください。** `call.id` だけを キーにすると、別のお客様の通話と取り違える可能性があります

`call`

キー	型	必須	説明
`id`	string	✓	通話 ID。Denwaban の管理画面の通話履歴と一致します
`status`	string	✓	通話の終了状態。`completed` など (5-3)
`startedAt`	string	–	通話開始日時 (ISO 8601)。**通話開始の情報を受け取れなかった場合は含まれません**
`endedAt`	string	–	通話終了日時 (ISO 8601)
`durationSeconds`	number	–	通話秒数
`callerNumber`	string	–	発信者の電話番号。**先頭に `+` が付かない形式** (例: `819012345678`)。**非通知の場合は `unknown`** が入ります
`calledNumber`	string	–	着信した番号。同じく先頭に `+` が付きません
`transferLabel`	string	–	転送先のラベル。**転送が発生した場合のみ**含まれます
`pointsConsumed`	number	–	消費ポイント

> **日時の形式**: 本書に出てくる日時は、`call.startedAt` / `call.endedAt` / `turns[].at` を 含めて**すべて `Z` または明示的な UTC オフセットが付きます**。オフセット無しのローカル 日時は送信しません。
>
> **電話番号の形式に注意**: `+` から始まる E.164 形式ではありません。`+` を前提に解析すると 失敗します。また非通知の通話では番号ではなく `unknown` という文字列が入るため、 電話番号として扱う前に判定してください。

`transcript`

キー	型	必須	説明
`truncated`	boolean	✓	**サイズ超過で文字起こしを削った場合に `true`** (1 MB を超える場合)
`text`	string	✓	**受信できた発話**を 1 つの文字列に連結したもの (`話者: 発言` を改行で連結)。管理画面の通話履歴と同じ形式です。**通話の全発話が揃っているとは限りません** (下記)。発話が無い場合は空文字
`turns`	array	✓	発話単位の配列。構造化して扱いたい場合に使用してください。発話が無い場合は空配列
`turns[].seq`	number	✓	通話内での通し番号。**昇順に並べ替えて**お使いください。**連続した値とは限りません** (欠番があっても異常ではありません)
`turns[].speaker`	string	✓	`ai` または `caller`
`turns[].text`	string	✓	発言内容
`turns[].at`	string	–	発言日時 (ISO 8601)

文字起こしが欠ける場合があります

1 通話あたりに受け取れる文字起こしの件数には上限があります。**5 分を超える長い通話では、 発話の一部が届かない場合があります。**

欠けるのは主に通話の終盤の発話です。 通話ログを早く確実にお届けするため、通話終了の 知らせが未処理の発話より優先して処理されます。その結果、処理しきれなかった終盤の発話が 間に合わないことがあります。混み合っているときは、5 分未満の通話でも起こり得ます。

この欠落は `transcript.truncated` では分かりません。 発話が届かなかったのか、もともと その発話が無かったのかを、ツール側から区別できないためです (`truncated` は サイズ超過で削った場合にのみ `true` になります)。

`turns[].seq` の欠番も判断材料にはなりません。**seq は同じ通話を購読している他のツールとも 共有した通し番号**のため、正常時から欠番が生じます。

文字起こしを議事録や証跡として使う場合は、文字起こしだけに依存しないでください。 録音 (6 章) が最も確実な記録です。

`recording`

キー	型	必須	説明
`status`	string	✓	録音の状態 (6-3)

キー	型	必須	説明
`url`	string | null	✓	ダウンロード URL。取得できない場合は `null`
`expiresAt`	string | null	✓	URL の有効期限 (ISO 8601)。**`Z` または明示的な UTC オフセットが必ず付きます。** `url` が `null` のときは `null`
`contentType`	string | null	✓	現在は `audio/mpeg` (MP3)

5-3. `call.status` の値

値	意味
`completed`	AI 受付で完結して終了した
`transferred`	AI の判断で転送先に取り次いだ
`transferred-by-vendor`	連携ツール側の操作 (強制取次) で転送した
`failed`	通話の確立や処理に失敗した

転送された通話は `completed` になりません。 通話が成立したかどうかで判定したい場合は、`completed` / `transferred` / `transferred-by-vendor` の 3 つをまとめて扱ってください。

転送が発生した場合は `call.transferLabel` に転送先のラベルが入ります。

> 今後値が追加される可能性があります。**未知の値が来ても落ちない**実装にしてください。

---

6. 録音の取得

6-1. 取得方法

`recording.url` に対し、**設定画面で登録したキーをベアラトークンとして付けて** GET してください。

> **キーをコマンドラインに書かないでください。** `-H "Authorization: Bearer <キー>"` の ような書き方は、**シェルの履歴に残り、実行中はプロセス一覧から他の利用者に読まれます** (環境変数を展開して渡した場合も、展開後の値が引数として見えます)。録音を取得できる 権限を持つ値なので、下の例のように**権限を絞った設定ファイル**から読み込むか、 Secret Manager 等から実行時に取得してください。

```
# キーは 600 の設定ファイルに置き、curl に読ませる (コマンドラインに実値を出さない)。
# **キーに含まれ得る " と \ をエスケープしてから書くこと。** キーには記号を含む
```

```
# 半角文字が使えるため、そのまま書くと curl が別のヘッダとして解釈するか、
# 設定ファイルの読み込み自体に失敗して認証できません
#   # **既存ファイルの権限は umask では直らない** (`>` は元の権限を引き継ぐ)。
#   # 600 で作り直してから書く
#   $ install -m 600 /dev/null ~/.denwaban-recording.curlrc
#   $ esc=$(printf '%s' "$KEY" | sed 's/[\\"]/\\&/g')
#   $ printf 'header = "Authorization: Bearer %s"\n' "$esc" > ~/.denwaban-recording.curlrc
CURL_CONFIG=~/.denwaban-recording.curlrc

# curl 8.4.0 以降が必要です。8.4.0 以降の --max-filesize は Content-Length の無い
# 応答でも転送中に上限を検出して中断します。それより前は受け取り終えるまで気づけません
MAX_BYTES=104857600   # 100 MB。実際の録音サイズに合わせて決めてください
tmp=$(mktemp)
trap 'rm -f "$tmp"' EXIT

# curl の終了コードを必ず見ること。時間切れ (28) でも %{http_code} には
# 受信済みヘッダの 200 が入り、一時ファイルには途中までの本文が残る
if ! out=$(curl -sS --max-time 300 --max-filesize "$MAX_BYTES" \
  --config "$CURL_CONFIG" \
  -o "$tmp" -w '%{http_code} %{size_download} %{content_type}' \
  "https://denwaban-tool-log-webhook-prod-cx44nrax2q-an.a.run.app/recordings/
        QmFzZTY0dXJsLWVuY29kZWQtb3BhcXVlLXRva2Vu"); then
  echo "取得に失敗しました (途中で切れた可能性があります)" >&2
  exit 1
fi
read -r code size ctype <<<"$out"

case "$code" in
  200)
    # 200 でも中身を検証してから音声として扱う。
    # メディア型はパラメータ (;charset=... 等) を落として、大小文字を無視して完全一致で見る
    # (前方一致だと audio/mpeg-malformed のような別の型まで通ってしまう)
    mediatype=$(printf '%s' "${ctype%%;*}" | tr -d '[:space:]' | tr '[:upper:]' '[:lower:]')
    [ "$mediatype" = "audio/mpeg" ] || { echo "音声ではありません: $ctype" >&2; exit 1; }
    [ "$size" -gt 0 ] && [ "$size" -le "$MAX_BYTES" ] || { echo "サイズが不正: $size" >&2;
        exit 1; }
    mv "$tmp" recording.mp3
    ;;
  # 404 は本文で pending / unavailable を見分ける ([6-2](#6-2-レスポンス))
  404) cat "$tmp" ;;
  *)   echo "failed: $code" >&2; exit 1 ;;
esac
```

上限そのものは `--max-filesize` が転送中に打ち切ります。`%{size_download}` の確認は、**成功した応答に対する追加の検証**です (受け取った実バイト数が想定内かを、保存前にもう一度 確かめるためのものです)。

成功したときだけ音声ファイルとして扱ってください。 応答を直接 `recording.mp3` へ 書き出すと、**エラー応答の JSON がそのまま音声ファイルとして保存されます。**

一方で `curl -f` (`--fail`) は使わないでください。**エラー時の本文が捨てられ、`404` の `pending` と `unavailable` を区別できなくなります** (6-2)。 本文が必要なので、上の例のように一時ファイルとステータスコードで判定してください。

> `--fail-with-body` も、上の例にそのまま足さないでください。本文は残りますが **`4xx` で終了コード 22 を返す**ため、終了コードを先に確認する上の例では `404` の判定に進む前に抜けてしまいます。
>
> URL は `recording.url` の値をそのまま使ってください。**リダイレクトの自動追従 (`curl -L` など) は有効にしないでください。** 認証情報が意図しない宛先へ送られる恐れが あります。

- 成功すると `200` + `Content-Type: audio/mpeg` で MP3 が返ります
- **キーはお客様が設定画面で設定した値**です。Denwaban のログイン情報とは無関係です
- URL を知っているだけでは取得できません。必ずキーが必要です

> **録音取得用のキーは、Webhook 送信先の認証情報とは別の設定です。** 送信先の認証に `basic` / `header` / `query` を選んだ場合でも、**録音の取得は常にベアラ認証**であり、 設定画面の「録音リンクのキー」に登録した値を使います。送信先の認証情報とは混同しないで ください。

キーは設定画面の生成ボタンで作成することを推奨します。 自分で決める場合も、 **32 文字以上のランダムな文字列**にしてください (設定できるのは **24〜256 文字の半角英数字と 記号**です。空白や全角文字は、`Authorization` ヘッダに載せられないため登録できません)。URL は受信側のログや保存先に残るため、短く推測しやすいキーだと、URL を入手した第三者に通話音声を取得される恐れがあります。

なお、**認証に繰り返し失敗するアクセスは一時的に制限され、`429` と `Retry-After` を 返します。** キーの設定を誤ったまま再試行を繰り返すと制限にかかるため、`401` が返った 時点で設定を確認してください。

6-2. レスポンス

ステータス	意味	対応
`200`	成功。MP3 が返ります	–
`401`	キーが未指定または不一致	設定画面のキーと突き合わせてください
`404` `{"status":"pending"}`	録音がまだ準備できていない	時間を置いて再試行してください
`404` `{"status":"unavailable"}`	録音が存在しない	再試行しても取得できません
`429`	認証の失敗が続いたため一時的に制限されている	`Retry-After` 秒後に再試行してください
`410`	**有効期限が切れている**、または**連携が解除・停止された**	再試行しても取得できません
`503`	Denwaban からの録音の取り出しに一時的に失敗した	`Retry-After` 秒後に再試行してください

`410` と `404` `{"status":"unavailable"}` は終端です。 再試行せず、録音は取得できな かったものとして確定してください。`404` `{"status":"pending"}` のみ、時間を置いた再試行に 意味があります。

連携が解除・停止された場合も `410` を返します (6-5)。 **`403` は返しません** (`410` に変換します)。上の表以外のステータスは返しません。

エラー応答の本文は JSON (`Content-Type: application/json`) で `{"status": "..."}` の形式です。 **本文を見て判断が変わるのは `404` だけ**です (`pending` / `unavailable`)。それ以外は ステータスコードだけで判断してください。

設定画面でキーを変更した場合、旧キーでのアクセスは `401` を返します。 リンク自体は 失効していないため、**新しいキーで取得すれば成功します。** `410` (終端) とは扱いが異なります。

`404` `{"status":"pending"}` は、 **`recording.status` が `available` のリンクにアクセスした ときだけ**返ります。送信の直後に録音の実体がまだ書き込まれていない場合に起こる一時的な状態です。

6-3. `recording.status` の値

値	url	意味
`available`	URL	録音が有効な通話です。リンクから取得できます (準備中の場合は `404` `{"status":"pending"}` が返ります → 6-2)
`pending`	`null`	送信時点でリンクを発行できませんでした。**後から準備される可能性があります。**お客様が**録音リンクのキーを未設定**の場合 (リンクを発行しても取得できないため) などに返ります
`unavailable`	`null`	録音が存在しません (録音が無効、または録音に失敗)

> `pending` で届いた通話の録音を後から取得する手段は、現時点では提供していません。 Denwaban の管理画面から通話履歴を確認してください。
>
> **`pending` / `unavailable` で届いた配信は、通話ログを保存したら「録音なし」で完了させて ください。** `url` が `null` で取得先が無く、後から取得する手段も提供していないため、 取得待ちとして残しても永久に処理が終わりません。 8-2 の再試行は、 **`available` のリンクへアクセスして `404` `{"status":"pending"}` が返った場合のみ**行ってください。

6-4. リンクのホスト (許可リストに登録してください)

録音リンクのホストは**次の 1 つに固定**されています。

```
denwaban-tool-log-webhook-prod-cx44nrax2q-an.a.run.app
```

キーを付ける前に `recording.url` を URL として解析し、次の 4 点をすべて確認してください。 1 つでも満たさない URL は、**キーを付けずに破棄してください** (理由は 8-2)。

確認する項目	期待する値
スキーム	**`https:` のみ**。`http:` は平文なので**キーが盗聴されます**
ホスト名	上記との**完全一致**
ポート	**既定 (443) のみ**。明示されている場合は `443` 以外を通さない
ユーザー情報	**含まれていないこと** (`https://user:pass@host/...` の形)

- **文字列の前方一致・後方一致で判定しないでください。** URL として解析してから 比較してください。`https://<許可ホスト>.example.net/` や `https://evil.example.net/?x=https://<許可ホスト>/` のような URL を通してしまいます
- ユーザー情報を見落とすと、`https://<許可ホスト>@evil.example.net/` を 「ホストが一致している」と誤判定する実装があります (`URL` で解析すれば `hostname` は `evil.example.net` になります)
- **ホストを変更する場合は事前にお知らせします。** 予告なく変わることはありません
- `recording.url` にはこのホストに続けて不透明なパスが入ります。パスの形式は 予告なく変わることがあるため、**パスの検証は行わないでください**
- リダイレクトの自動追従も無効にしてください (6-1)

6-5. 有効期限とアクセス可能な期間

録音リンクには以下の制限があります。**期限内に取得してください。**

- **発行から 30 日**で期限切れになります (`expiresAt` を参照)。期限切れ後は `410` を返します
- **ツールをアンインストールすると、期限内であってもリンクは無効になります**
- **ツールが停止状態になった場合も、期限内であってもリンクは無効になります。** 停止は お客様の操作のほか、**連携の異常が続いた場合に自動で行われることもあります**
- 設定画面で**キーを変更すると、既に発行済みのリンクも新しいキーでのみ**取得できるようになります

`expiresAt` はあくまで上限であり、それまでの取得を保証するものではありません。 録音が必要な場合は、**受信後できるだけ早く取得して自社側に保存してください。**

> **設計上の補足**: 本ツールは録音データを保持しません。リンクへのアクセス時に Denwaban から 取得して中継しています。そのためアンインストール後はリンクが機能しません。**長期保存が 必要な場合は、受信後にお客様側で保存してください。**

---

7. 重複と再送

7-1. 重複への対処 (受信側の責務)

同じ通話の Webhook が複数回届くことがあります。 ネットワークの問題や再送により発生します。

`X-Denwaban-Delivery` ヘッダ (= ペイロードの `deliveryId`) は**再送しても同じ値**です。 この値を一意キーとして保存し、**既に保存済みなら 2xx を返して何もしない**実装にしてください。

> **重複排除の印は、保存が成功してから付けてください。** 保存や処理を試みる前に「処理済み」と 記録すると、その後に失敗しても再送が弾かれてしまい、通話ログが失われます。 `deliveryId` に一意制約を付けた INSERT を使い、**重複エラーになったら受信済みとみなして 2xx を 返す**のが最も確実です。

重複判定には `deliveryId` を使ってください。`call.id` は使わないでください。

`call.id` は通話を識別する値で、配信を識別する値ではありません。複数の設定から配信を 受け取る場合、別のお客様の通話と値が衝突し、**まだ受け取っていない通話を「処理済み」と 誤判定して破棄する**恐れがあります。

- **配信の重複判定** → `deliveryId`
- **通話ログの保存キー** → `connectionId` + `call.id` (5-2)

用途が異なります。混同しないでください。

7-2. 再送の条件

応答	再送
2xx	しない (成功)
5xx	**する**
429 (Too Many Requests)	**する**
408 (Request Timeout)	**する**
タイムアウト・接続エラー	**する**
408 / 429 以外の 4xx	**しない**
3xx (リダイレクト)	**しない**

リダイレクトは追いません。 `301` / `302` / `307` / `308` が返った場合、その配信は 失敗として扱います。認証情報を意図しない宛先へ送らないためです。**転送先の URL を 設定画面に直接登録してください。**

`408` と `429` は一時的な状態とみなして再送します。それ以外の 4xx は設定の誤りとみなし 再送しません。認証エラー (401/403) が続く場合は設定を見直してください。

再送される内容は最初と同じです

再送時の本文は、最初に送ったものと完全に同じです。 再送のたびに作り直すことはありません。

- `occurredAt` も `recording` も、**最初の送信時点の内容のまま**です

- したがって、最初の送信時に `recording.status` が `pending` だった通話は、**再送されても `pending` のまま**です。録音リンクが後から付くことはありません
- 受信側は「最初の 1 通を保存し、以降は捨てる」処理で問題ありません。**後の配信に新しい情報が 含まれていることはありません**

ただし**本文以外は、そのときの設定に従います。** 送信先の URL や認証情報を設定画面で変更すると、 **まだ再送が残っている配信も、新しい URL・新しい認証情報で送信されます。** 認証情報を 入れ替えた場合、古い認証情報が使われ続けることはありません。

> **受信側が過負荷のときは `429` または 5xx を返してください。** `400` などを返すと再送されず、 その通話ログは失われます。

7-3. 再送の間隔

失敗するたびに間隔を広げながら、最大 5 回まで再送します。すべて失敗した場合は配信を打ち切り、 記録を残します。

再送対象の応答 (`429` / `503`) に `Retry-After` が付いている場合は、その時間を待ってから 再送します。 指定が通常の間隔より長い場合は指定に従います (指定を待たずに試行回数を 使い切って通話ログを捨てることはありません)。

- 受理する形式は **秒数 (delay-seconds) と HTTP-date の両方**です。解釈できない値や **過去の日時**が指定された場合は、指定が無かったものとして通常の間隔で再送します
- 尊重する上限は **15 分**です。これを超える指定は 15 分に丸めます
- **`Retry-After` に従って待った再送も、5 回のうちの 1 回**として数えます
- 15 分より長い復旧時間が必要な場合は、`429` を返し続けるのではなく**受信側で受け取って 保存だけ行い、処理を後回しにする**ことをおすすめします (8 章)

---

8. 受信側の実装

受信の入り口では「保存するだけ」にし、実際の処理は別で行うのが要点です。

8-1. 受信ハンドラ

Node.js (Express) での最小構成です。

TLS の終端は別途用意してください。 下の例は平文の HTTP で待ち受けます。送信先として 登録できるのは **https の URL のみ**なので、TLS を終端するリバースプロキシ (ロードバランサ等) の背後で動かすか、`https.createServer` で待ち受けるように置き換えてください。

```
const express = require("express");
const { Pool } = require("pg");
const app = express();
```

```
// 保存先。**接続の取得とクエリの両方に期限を設ける** (下の 3. を参照)
const db = new Pool({
  connectionString: process.env.DATABASE_URL,
  // プールが枯渇したときに接続の取得待ちで止まらないようにする
  connectionTimeoutMillis: 3_000,
  // クライアント側でクエリを打ち切る
  query_timeout: 5_000,
  // サーバ側でも打ち切る (保険。接続の取得待ちには効かない)
  statement_timeout: 5_000,
});

// **待機中の接続で起きたエラーを必ず受けること。** リスナーが無いと Node は
// 未処理の error イベントとして扱い、**プロセスごと落ちます** (DB の再起動や
// フェイルオーバーで起こります)。落ちると 5xx すら返せず、再送の機会も失います
db.on("error", (err) => {
  console.error("database pool error", err);
});

const EXPECTED_TOKEN = process.env.DENWABAN_WEBHOOK_TOKEN;

// 未設定のまま起動しないこと。undefined のままだと
// `Authorization: Bearer undefined` で認証を通過してしまう
if (!EXPECTED_TOKEN) {
  throw new Error("DENWABAN_WEBHOOK_TOKEN が設定されていません");
}

// 1. 認証 (設定画面で bearer を選んだ場合)
//    本文の解析より前に置くこと。app.use(express.json()) を全体にかけると、
//    認証を知らない相手にも最大 1 MB の JSON を解析させることになる
function authenticate(req, res, next) {
  const auth = req.headers.authorization || "";
  if (auth !== `Bearer ${EXPECTED_TOKEN}`) {
    return res.sendStatus(401);
  }
  next();
}

// 送信側の最大ボディサイズと同じ 1 MB。これを下回ると長い通話が 413 で失われる
const parseJson = express.json({ limit: "1mb" });

app.post("/denwaban/calls", authenticate, parseJson, async (req, res) => {
  // 2. 配信 ID を検証する。ヘッダが欠けたまま保存すると UNIQUE 制約が効かず
  //    (NULL は重複と判定されない)、再送のたびに同じ通話を処理してしまう
  const deliveryId = req.headers["x-denwaban-delivery"];
  if (
    typeof deliveryId !== "string" ||
    deliveryId === "" ||
    deliveryId !== req.body?.deliveryId
  ) {
    console.error("invalid delivery id", { deliveryId });
    return res.sendStatus(400);
  }

  // 3. 受信内容をそのまま保存する。ここまでが同期処理
  //    deliveries テーブルの delivery_id に UNIQUE 制約を張っておく
  //    保存の呼び出し全体に 10 秒未満の締め切りを設けること (上の Pool の設定で
  //    最大 3 + 5 = 8 秒)。接続の取得待ち (プール枯渇) や送受信の停止は
```

```
  //    statement_timeout では止まらないので、クライアント側の期限で打ち切り、
  //    DB 側の statement_timeout は保険として併用する
  try {
    await db.query(
      `INSERT INTO deliveries (delivery_id, payload, status)
       VALUES ($1, $2, 'received')
       ON CONFLICT (delivery_id) DO NOTHING`,
      [deliveryId, req.body],
    );
  } catch (err) {
    // 保存できなかった → 5xx を返して再送してもらう
    console.error("failed to persist delivery", err);
    return res.sendStatus(500);
  }

  // 4. 保存できたので 2xx。重複していた場合もここに来る（成功として返す）
  return res.sendStatus(200);
});

// TLS を終端するリバースプロキシの背後で待ち受ける想定。
// このプロセスを直接インターネットに公開しないこと（送信先は https のみ）
app.listen(3000);
```

> `400` **を返した配信は再送されません** (7-2)。配信 ID が届かないのは 経路上のプロキシがヘッダを削っているなどの構成の問題です。**発生をログに残して検知できる ようにし、原因を取り除いてください。**

保存した配信は、**受信とは別の経路** (定期実行するワーカーなど) で処理します。失敗しても 保存内容は残っているため、やり直せます。

この部分は各社の基盤に依存するため、**満たすべき要件**のみを示します。

8-2. 保存した配信を処理する側の要件

取り出し

- 1 回の実行で取り出す件数に**上限を設ける**
- **ワーカーを複数動かす場合は、配信を排他的に確保してから処理する。** 単に「未処理」で 絞り込むだけでは、同じ配信を複数のワーカーが同時に処理します
- **処理中にプロセスが落ちた配信を、一定時間後に未処理へ戻す**仕組みを用意する

保存 (通話ログ・録音とも)

再送や再試行で、**同じ配信を複数回処理します。** 保存はすべて冪等にしてください。

- **通話ログは** `connectionId` + `call.id` **をキーにした upsert にする。** 追記型にすると 重複します
- **録音の保存も同様に冪等にする。** 録音を保存した直後・完了を記録する前に処理が中断すると、 同じ配信が再処理されます。追記型のストレージや添付 API を使う場合、同じ録音が二重に 保存されます

録音の取得

`recording.status` が `available` のものだけ取得します。

`pending` / `unavailable` の配信は取得を行わず、通話ログを保存したら「録音なし」として 完了させてください。 取得先の URL が無く、後から取得する手段も提供していないため (6-3)、未完了のまま残すとワーカーが同じ配信を取り直し続けます。

> **キーを送る前に、URL のホストを検証してください。** payload の `recording.url` を無条件に 信頼して取得すると、**細工された配信によって録音キーを第三者のホストへ送ってしまう**恐れが あります (送信先に認証を設定していない場合や、認証情報が漏れた場合)。
>
> ホストは 6-4 の 1 つに固定されています。 **URL として解析し、スキーム (`https:`)・ホスト名の完全一致・ポート (443)・ ユーザー情報が無いことの 4 点を確認**してください。1 つでも満たさない URL には キーを付けずに破棄してください。とくに `http:` を受け入れると、**録音キーが平文で 流れます**。リダイレクトも追わないでください。
>
> **必ずタイムアウトを設定してください。** 接続後に応答が止まると、多くの HTTP クライアントは 既定では待ち続けます。1 件の取得が固まると、その配信が処理中のまま滞留します。
>
> **`200` でも中身を検証してから保存してください。** `Content-Type` が `audio/mpeg` で あることを確認し、**保存するサイズに上限を設けて、超えたら打ち切って破棄**してください (`Content-Length` は自己申告なので、**実際に受信したバイト数**でも数えてください)。 上限を設けずにストリームを書き出すと、想定外の応答でディスクやメモリを使い切ります。

結果によって扱いが分かれます。

結果	扱い
`200`	保存して**完了**
`410` (期限切れ)	取得できないので**完了**にする (録音なしで確定)
`404` + `{"status":"unavailable"}`	録音が存在しないので**完了**にする
`404` + `{"status":"pending"}`	まだ準備中。**再試行する**
`401`	**設定の誤り。すぐに再試行せず、担当者へ通知して取得を止める**
`429`	一時的な制限。**`Retry-After` 秒待ってから再試行する**
`5xx`	一時的な障害の可能性があるので**再試行する**
接続エラー・タイムアウト	**再試行する**

`401` が返ったら、同じキーで再試行しても回復しません。 繰り返すと 6-1 の 制限にかかり、担当者がキーを直す前に再試行の上限を使い切って録音を失います。**取得を止めて 通知**し、キーを直してから再開してください。

`404` **は本文を見て判断してください。** `pending` と `unavailable` で扱いが逆になります。 また接続エラーやタイムアウトは応答ではなく例外として発生するため、捕捉しないと再試行の 記録に進まないまま処理全体が止まります。

再試行

- **再試行の間隔を空ける。** `pending` は録音の準備が終わっていない状態です。**最低でも 30 秒**空け、以降は間隔を広げてください (指数バックオフ)。間隔を空けずに繰り返すと、 録音が準備される前に試行回数を使い切り、**本来取得できた録音を失います**
- **試行回数に上限を設ける。** 上限が無いと、取得できない録音を永久に叩き続けます
- **上限に達した配信は、未処理とは区別できる状態にする。** 未処理のまま残すと、二度と 処理されないのに「処理待ち」に見え、監視から漏れます
- **その状態の発生を監視し、手動で対処できるようにしてください。** 通話ログがそのままでは 失われます

全体

- **1 件の失敗で、他の配信の処理を止めない**

---

9. 実装時のチェックリスト

受信側を実装する際は、以下を確認してください。

受信と応答

- [ ] **受信内容を保存してから 2xx を返している** (保存前に 2xx を返していない)
- [ ] 保存先は**プロセスが落ちても残るもの**である (メモリ上のキューではない)
- [ ] **保存に失敗したら 5xx を返している** (再送してもらうため)
- [ ] **過負荷時に `429` か 5xx を返している** (`400` などを返すと再送されず失われる)
- [ ] **ボディサイズの上限を 1 MB 以上**にしている (下回ると長い通話が `413` で失われる)
- [ ] **2xx を 10 秒以内に返している** (重い処理は保存後に非同期で行う)
- [ ] **保存先の呼び出しにも 10 秒未満のタイムアウト**を設定している (応答不能時に再送が積み上がらないように)
- [ ] **認証を本文の解析より前に行っている** (未認証の相手に大きな JSON を解析させない)
- [ ]

`X-Denwaban-Delivery` **が空でなく、ペイロードの** `deliveryId` **と一致することを確認している**

- [ ] `deliveryId` **で重複を排除している。**ただし**保存が成功してから**印を付けている。`call.id` **は重複判定に使わない**
- [ ] **リダイレクトを返していない** (3xx は追わないため配信が失敗します)

データの扱い

- [ ] `connectionId` **と組み合わせて保存している** (複数の設定から受け取る場合、`call.id` だけでは取り違える)
- [ ] `turns` を `seq` **で並べ替えて**いる
- [ ] `transcript.truncated` **を確認している** (証跡として使う場合)。ただし `false` **でも発話が欠けていることがある**点を理解している
- [ ] `turns[].seq` **の欠番を異常として扱っていない** (他ツールと共有の通し番号のため、正常時から欠番が出ます)
- [ ] `summary` が `null` でも落ちない
- [ ] `callerNumber` **が** `unknown` **(非通知) でも落ちない。**また `+` **が付かない形式**として扱っている
- [ ] `call.startedAt` / `callerNumber` **が無くても落ちない** (通話開始の情報を受け取れなかった場合は含まれません)
- [ ] `call.status` **の** `transferred` / `transferred-by-vendor` **を考慮している** (転送された通話は `completed` にならない)
- [ ] `call.status` に**未知の値**が来ても落ちない

録音

- [ ] `recording.status` が `pending` / `unavailable` でも落ちない
- [ ] `pending` / `unavailable` **の配信を「録音なし」で完了させている** (取得待ちのまま滞留させない)
- [ ] **キー変更後の** `401` **と、失効を表す** `410` **を区別している** (`401` は新しいキーで取得できる)
- [ ] **録音を受信後できるだけ早く取得している。**長期保存が必要なら自社側に保存する
- [ ]

録音取得に**送信先の認証情報ではなく、録音リンク用のキー**を使っている

- [ ] **キーを送る前に URL のホストを検証している。**リダイレクトも追わない
- [ ] **ダウンロードにタイムアウトを設定している**
- [ ] **`200` の応答も検証している** (`Content-Type` が `audio/mpeg`、受信サイズに上限を設けて超えたら破棄)
- [ ] **エラー応答の本文を音声ファイルとして保存していない** (`404` の本文は `pending` / `unavailable` の判定に使う)
- [ ] **`401` で再試行を続けていない** (通知して止める)

---

10. 個人情報の取扱いについて

送信されるデータには、**通話の文字起こしと要約、発信者の電話番号**が含まれます。これらには 氏名・住所などの個人情報が含まれる可能性があります。

録音の音声そのものは送信しません。 送信されるのは認証付きのダウンロードリンクだけです。 ただし、**そのリンクから音声を取得できる**ため、リンクとキーの管理も音声データと同等に 扱ってください。取得した音声を保存する場合、その保管もお客様の責任範囲になります。

- 送信先の URL とその保管環境は、お客様の責任で管理してください
- 認証を設定していない送信先には送信しないことを強く推奨します
- 受信したデータの保存期間・削除方針は、お客様のプライバシーポリシーに従ってください

---

11. 変更履歴

バージョン	日付	内容
0.1.0 (確定版・改訂)	2026-08-14	**録音リンクのホストを明記** (6-4)。受信側が許可リストを構成できるようにするためで、例示用ドメインを実際のホストに置き換えた。あわせて概要 (1 章) に「録音が有効でも `recording.url` が `null` になる場合がある」ことを追記し、受信側の実装例に保存先の初期化とタイムアウト・TLS 終端の前提・録音キーをコマンドラインに出さない取得方法を反映
0.1.0 (確定版)	2026-08-13	**実装完了。全項目を実機で突き合わせ済み** (実着信での配信・録音の取得・認証失敗時の応答まで確認)。ペイロードの例を実際の送信内容に合わせた (日時は `Z` 付き、`turns[].seq` は欠番のある実例)。`recording.status` が `pending` になる具体的な条件を追記

バージョン	日付	内容
0.1.0 (改訂)	2026-08-13	録音の扱いを実装に合わせて明確化: **送信は録音の準備完了を待たない** (録音が有効な通話には `available` のリンクを付けて送信し、準備中は取得時に `404` `{"status":"pending"}`)。録音の取得で `503` (一時障害) が返り得ることを追記
0.1.0 (改訂)	2026-08-13	要約の待ち時間に上限 (30 秒) があることを追記。超過時は `summary: null` で送信される
0.1.0 (改訂)	2026-08-12	文字起こしが欠ける箇所と条件を明確化 (主に通話の終盤 / 混雑時は短い通話でも発生)。`transcript.text` の説明を「受信できた発話の連結」に修正
0.1.0	2026-08-10	初版 (ドラフト)