Send Session Data to Your System via Webhook

Spokiva Webhook Integration Guide

When a voice session (call) ends, Spokiva can send all of that session's data to your own system through a webhook. You get the recording, the transcript, the conversation context, and usage details — everything shown on the **Workflow Runs** page — delivered as a single JSON payload to a URL you choose.

This guide is written for the developer who will receive and process these webhooks.

How it works

- After a session completes, Spokiva fires an HTTP `POST` to the webhook URL you configure on the workflow.
- The payload is a JSON object containing the full session record: recording URLs, transcript URL, gathered context, usage, and more.
- Delivery is durable: Spokiva retries transient failures (network errors, server 5xx, timeouts) with backoff, and stops after the configured attempt limit.
- Each delivery carries a stable `X-Spokiva-Delivery-Id` header so your system can deduplicate retried deliveries.

What the payload contains

The webhook payload is a JSON object with the following fields:

- **`event`** : Always `"session_completed"`.
- **`workflow_run_id`** : The unique id of this session run.
- **`workflow_id`** : The id of the workflow (voice agent).
- **`workflow_name`** : The name of the workflow.
- **`call_time`** : When the session ran (ISO 8601).
- **`call_disposition`** : How the session ended (e.g. `user_hangup`).
- **`call_id`** : The provider call id.
- **`call_status`** : The session status.
- **`caller_name`** : The caller's name, if captured.
- **`caller_type`** : The caller type, if captured.
- **`extracted_variables`** : Structured data extracted from the conversation.
- **`initial_context`** : Visitor context passed in at the start (page URL, etc.).
- **`gathered_context`** : Everything gathered during the conversation.
- **`usage_info`** : Usage stats: LLM tokens, TTS/STT character counts, duration.
- **`annotations`** : QA / integration annotations, if any.

- `recording_url` : The full session recording (downloadable).
- `transcript_url` : The full transcript (downloadable).
- `user_recording_url` : The caller's audio track.
- `bot_recording_url` : The agent's audio track.

> *`recording_url` , `transcript_url` , `user_recording_url` , and `bot_recording_url` are temporary signed download links. Fetch them promptly (within minutes) — they expire.*

Configuring the webhook

The webhook is configured on the workflow in the Spokiva admin panel:

1. Log in to https://admin.spokiva.neptuneinfotech.com.
2. Open the workflow (voice agent).
3. In the workflow canvas, find the **Webhook** node (named "Session Completion Webhook"). If it is not present, ask your Spokiva representative to add it.
4. In the node, set: - **Endpoint URL**: the URL on your server that will receive the `POST` . - **HTTP Method**: `POST` . - Leave **Enabled** on.
5. Save and publish the workflow.

No code is needed on the widget side — the webhook fires automatically when a session ends.

Receiving the webhook

Spokiva sends an HTTP `POST` with:

- **Content-Type**: `application/json`
- **Body**: the session payload described above.

The webhook also includes these headers so you can deduplicate and trace:

- `X-Spokiva-Delivery-Id` : Stable id for this delivery. Same value on retries.
- `X-Spokiva-Workflow-Run-Id` : The session run id.
- `X-Spokiva-Delivery-Attempt` : Which attempt this is (1, 2, 3, ...).

Your endpoint should respond with HTTP `2xx` to confirm receipt. A non-2xx or a timeout is treated as a failure and retried.

Retries

Transient failures are retried with exponential backoff:

- HTTP statuses that are retried: `408` , `425` , `429` , `500` , `502` , `503` , `504` , plus connection / DNS / timeout errors.
- Permanent failures (most other `4xx` , e.g. auth or not-found) are **not** retried — fix the URL and trigger a new session to test.

- The default attempt limit is **5**, with backoff capped at **10 minutes**.
- If all attempts fail, the delivery is parked as a dead letter for inspection (visible in the admin panel).

Because retries can happen, **make your webhook handler idempotent** using `X-Spokiva-Delivery-Id`: store the id of deliveries you've already processed and skip duplicates.

Example payload


```
{
  "event": "session_completed",
  "workflow_run_id": "1306",
  "workflow_id": "36",
  "workflow_name": "Support Agent",
  "call_time": "2026-09-18T11:35:57.711649+00:00",
  "call_disposition": "user_hangup",
  "call_id": "06b43b50-6882-486d-b22f-8c301bd7734c",
  "call_status": "user_hangup",
  "caller_name": "",
  "caller_type": "driver",
  "extracted_variables": "{\"caller_name\": null}",
  "initial_context": "{\"source\": \"widget\", \"page_url\": \"https://client-site.com\"}",
  "gathered_context": "{\"call_id\": \"06b43b50-...\", \"call_status\": \"user_hangup\",
\"call_disposition\": \"user_hangup\", \"call_tags\": [\"user_hangup\", \"user_speech\"]}",
  "usage_info": "{\"llm\": {\"OpenAILLMService#21|||gpt-4.1\": {\"prompt_tokens\": 3761}},
\"tts\": {\"SarvamTTSService#21|||bulbul:v3\": 386}, \"call_duration_seconds\": 50}",
  "annotations": "{}",
  "recording_url":
"https://call.neptuneinfotech.com/api/v1/public/download/workflow/abc123/recording",
  "transcript_url":
"https://call.neptuneinfotech.com/api/v1/public/download/workflow/abc123/transcript",
  "user_recording_url":
"https://call.neptuneinfotech.com/api/v1/public/download/workflow/abc123/user_recording",
  "bot_recording_url":
"https://call.neptuneinfotech.com/api/v1/public/download/workflow/abc123/bot_recording"
}
```


> *The nested objects (`initial_context`, `gathered_context`, `usage_info`, `annotations`, `extracted_variables`) are delivered as JSON strings inside the payload. Parse them with `JSON.parse` before use.*

Example receiver (Node.js)

```
 const http = require('http');

http.createServer((req, res) => {
  if (req.method === 'POST') {
    let body = '';
    req.on('data', (c) => body += c);
    req.on('end', () => {
      const deliveryId = req.headers['x-spokiva-delivery-id'];
      console.log('delivery', deliveryId, 'payload', body);
      // TODO: store deliveryId to deduplicate; process the payload.
      res.writeHead(200, { 'Content-Type': 'application/json' });
      res.end(JSON.stringify({ ok: true }));
    });
  } else {
    res.writeHead(200); res.end();
  }
}).listen(9000, () => console.log('webhook receiver on :9000'));
```

Testing

To test the webhook, run a real session against the workflow (for example via the widget or a test call). When it ends, watch your webhook receiver for the `POST`. Check the **Workflow Runs** page in the admin panel to confirm the session data matches.

If you don't receive anything:

- Confirm the workflow is **published** with the webhook node **enabled**.
- Confirm the **Endpoint URL** is reachable from the internet (not `localhost`), and responds `2xx` promptly.
- Check the admin panel for dead-lettered deliveries.

Support

For help, contact your Spokiva representative or reach out through the Neptune Infotech team. Include the workflow name and the session run id.