Forms Logic

Webhook Integration Guide

A Plain-Language Guide for Getting Started with Automated Workflow Events

What Is a Webhook?

Think of a webhook as an automatic text alert. Instead of you having to log in to Forms Logic and manually check whether a workflow has been signed, approved, or finalized, Forms Logic sends your other system a notification the moment it happens — automatically, in real time.

Here is a simple way to picture it:

> **Real-World Analogy**
> Imagine you order a package online. Instead of refreshing the tracking page every hour, you sign up for shipping alerts — and you get a text the moment your package ships, then again when it is delivered. A webhook works the same way. You register once, and Forms Logic sends your system an alert every time a key event happens in a workflow.

Without webhooks, your team would need to manually monitor Forms Logic and copy information over to other systems. With webhooks, that work happens automatically in the background.

Why Would You Use Webhooks?

Webhooks are useful whenever your organization wants other software systems (a CRM, a back-office platform, a database, etc.) to stay in sync with what is happening inside Forms Logic. Common use cases include:

- Automatically updating a client record in your CRM when a workflow is finalized
- Triggering a back-office process the moment a client signs documents
- Logging workflow activity in your own internal system for compliance or recordkeeping
- Sending a notification to your team when a workflow reaches a key stage like approval or completion

The key benefit is speed and accuracy — your systems stay current without anyone having to manually re-enter information.

How Do Webhooks Work?

Here is the step-by-step flow of how a webhook works between Forms Logic and your system:

Step	What Happens
1. Event occurs	Something happens in Forms Logic — a workflow is signed, finalized, approved, etc.
2. Forms Logic sends a notification	Forms Logic automatically packages up the details of that event and sends them to your system.
3. Your system receives the notification	Your endpoint (a designated address at your organization) receives the message.
4. Your system takes action	Your system reads the notification and does something with it — updates a record, sends an alert, logs a transaction, etc.
5. Your system confirms receipt	Your system sends back a quick confirmation ("got it") so Forms Logic knows the message was delivered.

> **Why respond quickly?**
> Your system should send that confirmation back to Forms Logic as fast as possible — within a second or two, before doing anything else with the data. This prevents timeouts. Think of it like signing for a package before you open it. Best practice is to store the notification first, then process it separately.

What Information Does Forms Logic Send?

Every notification that Forms Logic sends includes a structured package of information. Here is what is included and what each piece means:

The Outer Envelope

Every notification is wrapped in an outer envelope that provides basic context:

Field	What It Means
id	A unique ID for this specific notification event.
event	What happened — for example: workflow_finalized or workflow_signed.
apiVersion	The version of the Forms Logic system that sent this notification.
created	The date and time the event occurred (used to put events in order).
type	The kind of data being sent — usually “workflow” for workflow-related events.
data	The full details of the event — see below.

Workflow Details

The data section contains rich details about the specific workflow that triggered the event:

Field	What It Means
id	Unique ID for the workflow.
organization	Your organization’s name.
activityType	The type of activity — for example, New Account.
transactionType	The type of transaction — for example, New Business Advisory/RIA Brokerage.
accountType	The account type — for example, Rollover IRA.
custodian	The custodian involved — for example, Fidelity or Pershing.
state	The current status of the workflow — for example, Finalized or Signed.
transactionAmount	The dollar amount of the transaction.
signatureType	How the documents were signed — for example, DocuSign.
repCodeChoosen	The rep code used to create this workflow.
forms	A list of forms included in the workflow and whether each has been signed.
docs	A list of documents attached to the workflow — submitted, signed, or printed.

Advisor Details

Information about the advisor associated with the workflow is included automatically:

Field	What It Means
id / providerKey	The advisor's unique identifier in Forms Logic and in your SSO/login system.
name fields	First name, last name, middle name, prefix, suffix, and full name.
contact	Email addresses and phone numbers (home, work, and mobile).
addresses	Home and work address.
repCode	The advisor's representative code.
createdDate / createdBy	When and by whom the advisor record was originally created.

Client Details

Information about the client associated with the workflow is also included:

Field	What It Means
id / providerKey	The client's unique identifier in Forms Logic and in your CRM.
clientType	Whether the client is an Individual or an Entity (such as a trust or business).
name fields	First name, last name, middle name, prefix, suffix, full name, and entity name.
contact	Email addresses and phone numbers.
addresses	Home and work address.
financial info	Net worth, liquid net worth, annual income, tax bracket (if available).
investment profile	Risk tolerance, portfolio objective, time horizon, and source of funds (if available).
citizenship / employment	Citizenship status, employer, occupation (if available).

Document Details

Each document attached to the workflow is listed separately with:

Field	What It Means
id	Unique ID for the document.
fileName	The name of the file.
category	What kind of document it is — see the table below.
formType	The file format: PDF or DOC.

Document categories explained:

Category	What It Means
Submitted Document	A document that was submitted for signing.
Signed Document	A document that has been fully signed.
Printed Document	A document that was requested for printing rather than electronic signing.
Audit Report	A system-generated report tracking the workflow activity.

Custom Fields

Forms Logic also supports custom fields — any additional data points your organization has configured specifically for your workflows. Each custom field includes a name and a value.

How to Set Up Webhooks

Setting up webhooks involves three main steps. Your IT team or technology partner will handle the technical implementation — this section gives you a clear picture of what is involved so you can coordinate effectively.

Step 1: Create a Webhook Handler (Your IT Team)

Your IT team needs to set up a destination address — called an endpoint — where Forms Logic can send notifications. Think of this as a designated inbox that only receives messages from Forms Logic.

Your endpoint needs to:

- Be accessible via a web address (a URL) that Forms Logic can reach

- Accept incoming messages sent via the POST method (a standard way of receiving data over the internet)
- Send back a quick confirmation ("got it" response) immediately upon receiving a message

> **For your IT team: Testing tip**
> During development and testing, your team can use a tool called ngrok (ngrok.com) to create a temporary public address for a local server, making it easy to test webhook delivery before going live.

Step 2: Secure Your Webhook (Your IT Team)

Because webhooks carry sensitive financial data, Forms Logic uses encryption to protect every notification it sends. Your IT team will need to implement two security verifications:

- **The contents of each notification are encrypted so only your system can read them. Think of it as a sealed envelope that only your office can open.** JSON Web Encryption (JWE):
- **Every notification is digitally signed by Forms Logic so your system can verify the message genuinely came from Forms Logic and was not altered in transit. Think of it as a tamper-evident seal.** JSON Web Signature (JWS):

> **Why does this matter?**
> These two layers of security ensure that (1) no one else can read the data being sent, and (2) you can confirm the message actually came from Forms Logic. Both protections are strongly recommended and should be implemented before going live.

Step 3: Register Your Endpoint in Forms Logic

Once your endpoint is built and secured, the final step is registering it inside Forms Logic. This tells Forms Logic where to send notifications. Here's how:

1. Log in to your Forms Logic account.
2. Go to Site Settings (typically found in the Setup or Admin area).
3. Enter your webhook endpoint URL (the address your IT team created).
4. Save the registration.

From that point forward, Forms Logic will automatically send event notifications to your registered endpoint whenever a qualifying event occurs.

What Events Trigger a Webhook?

Forms Logic will send a webhook notification when significant things happen in a workflow. Examples include:

Event	What It Means
workflow_finalized	A workflow has been fully completed and finalized.
workflow_signed	A workflow has been signed (may also trigger document-level events).
Workflow approved	A workflow has passed through an approval stage.
Document signed	A specific document within a workflow has been signed.
State change	A workflow has moved from one stage to another.

Each event sends the full snapshot of the workflow at that moment in time, so your system always has complete, current information.

Questions or Next Steps?

If you have questions about setting up webhooks for your Forms Logic account, or if you need help coordinating between your IT team and our platform, please reach out to your Forms Logic account representative.

If you are unsure whether webhooks are the right solution for your integration needs, your account team can walk you through the options and help you determine the best approach for your specific workflows.

TECHNICAL APPENDIX

Developer & IT Implementation Reference

This appendix is intended for developers and IT staff responsible for implementing the webhook integration. It contains the complete technical specifications, data schemas, security requirements, and endpoint setup guidelines.

A1 — Event Object Structure

Every webhook notification sent by Forms Logic is delivered as an HTTPS POST request. The payload is a JSON Web Token (JWT) that, once decrypted, contains the following top-level Event Object:

```
{
  "id":          "456asas1MqqbKLt4dXK03v5qaIbiNCC",
  "event":       "workflow_finalized",
  "apiVersion": "2024-04-01",
  "created":     16800645587,
  "type":        "workflow",
  "data":        { ... }
}
```

Field	Type / Description
id	String — Unique identifier for the event. Always treat as a string; do not cast to integer.
event	String — The triggering event name (e.g. workflow_finalized, workflow_signed).
apiVersion	String — API release date (e.g. 2024-04-01). Subject to change; version against this field.
created	Integer (UNIX Timestamp) — UTC epoch seconds. Use to order events and detect duplicates.
type	String — Data type of the payload object. Currently always “workflow”.
data	JSON Object — The full workflow snapshot. See A2 below.

> ⚠ **Important: ID Field Types**

> Never assume the data type of any ID field. Forms Logic reserves the right to change ID formats at any time. Always store and compare IDs as strings to avoid type-mismatch bugs in your integration.

A2 — Workflow Data Object

The data field contains a Workflow Data Object. Below is an annotated schema of all fields. Note: all field names are camelCased.

Workflow-Level Fields

Field	Type / Description
id	String — UUID for the workflow (e.g. c0f87fab-3398-4def-9279-1df84288c98g).
organization	String — Name of the organization that owns the workflow.
activityType	String — e.g. New-Account.
relationshipType	String — e.g. Advisory (Fee).
transactionType	String — e.g. New Business Advisory/RIA Brokerage.
accountType	String — e.g. Rollover IRA.
vendor	String | null — Vendor name if applicable.
custodian	String — Custodian name (e.g. Fidelity).
state	String — Current workflow state (e.g. Finalized, Signed).
transactionAmount	String — Formatted dollar amount (e.g. $60,000.00).
multipleTransactions	String — Aggregate value if multiple transactions exist.
signatureType	String — Signature method used (e.g. Docusign).
formZippedFilesUrl	String — URL to download a ZIP of all form files for this workflow.
aesKeyBase64	String — Base64-encoded AES key used to decrypt the zipped form files.
repCodeChoosen	String — The rep code selected when the workflow was created.

Advisor Object (data.advisor)

Field	Type / Description
id	Integer — Internal advisor ID.
providerKey	String — Email (user/password login) or SSO-provided external identifier.
prefix / suffix	String | null
firstName / middleName / lastName / fullName	String
gender	String | null
dob	String — Format: YYYY-MM-DD.
email / homeEmail / workEmail	String
homePhone / workPhone / mobilePhone	String — Format varies; normalize on your end.
home / work	Object — address1, address2, city, state, zip, country.
maritalStatus	String | null
repCode	String — Advisor's primary rep code.
ssn	String — Treat as PII; handle per your data security policies.
createdDate	String — ISO 8601 format (e.g. 2024-07-06T11:58:21.344564Z).
createdBy	String — Username or identifier of who created the record.

Client Object (data.client)

Field	Type / Description
id	Integer — Internal client ID.
providerKey	String — CRM-provided external identifier.
clientType	String — Individual or Entity.
prefix / suffix	String | null
firstName / middleName / lastName / fullName / entityName	String | null
gender / maritalStatus	String | null

dob	String | null — Format: YYYY-MM-DD.
trustDate	String | null — ISO 8601. Populated for trust accounts.
email / homeEmail / workEmail	String
homePhone / workPhone / mobilePhone	String
home / work	Object — address1, address2, city, state, zip, country.
occupation / employer / yearsWithCurrentEmployer	String | null
citizenship	String | null
repCodes	Array | null
ssn / taxId	String | null — PII; handle per your data security policies.
totalNetWorth / liquidNetWorth / annualIncome	Number | null
taxBracket / riskTolerance / portObjective / timeHorizon / sourceOfFunds	String | null
createdDate	String — ISO 8601.
createdBy	String

Forms Array (data.forms)

An array of form objects included in the workflow:

Field	Type / Description
id	String — Internal form ID.
quikFormId	String — Quik! platform form identifier.
productId	String — Product identifier.
formName	String — Display name of the form.
isSigned	Boolean — true if this specific form has been signed.
createdDate	String — ISO 8601.

createdBy	String

Docs Array (data.docs)

An array of documents attached to the workflow:

Field	Type / Description
id	String — UUID for the document.
fileName	String — Full file name including extension.
category	String — One of: AuditReport | SubmittedDocument | PrintedDocument | SignedDocument.
formType	String — Pdf or Doc.

Custom Fields Array (data.customFields)

An array of organization-defined key-value pairs:

Field	Type / Description
fieldName	String — Name of the custom field as configured in your Forms Logic account.
fieldValue	String — The value for this field on this workflow.

A3 — Security: JWE Encryption & JWS Signature

Forms Logic transmits all webhook payloads over HTTPS using JSON Web Encryption (JWE). You must also verify the JSON Web Signature (JWS) to authenticate that the message originated from Forms Logic. Both layers are strongly recommended before processing any payload in production.

JSON Web Encryption (JWE)

- The entire Event Object payload is encrypted using JWE before transmission.
- Your endpoint receives an encrypted token, not plaintext JSON.
- You must decrypt the token using the appropriate key before you can access the data.

- The aesKeyBase64 field in the Workflow Data Object contains the Base64-encoded AES key needed to decrypt the zipped form file downloads (separate from the JWE transport key).

JSON Web Signature (JWS) Verification

- Every payload is signed by Forms Logic using JWS.
- Before processing a webhook, verify the signature to confirm the message is authentic and has not been tampered with in transit.
- Reject any payload that fails signature verification.

> **Recommended Libraries**
> Most modern languages have well-maintained JWT/JWE/JWS libraries. For Node.js: jose. For Python: python-jose or joserfc. For Java: Nimbus JOSE+JWT. Always use a library — do not implement JWE/JWS crypto from scratch.

A4 — Endpoint Implementation Requirements

Requirement	Detail
Protocol	HTTPS required. HTTP endpoints will not be accepted in production.
Method	Must accept HTTP POST requests only.
Response code	Return HTTP 2xx immediately upon receipt. Any non-2xx is treated as a delivery failure.
Response time	Respond within a few seconds. Long-running logic must be handled asynchronously.
Async processing	Queue the raw encrypted payload immediately upon receipt; decrypt and process in a background worker. This also enables replay on bug fix.
Idempotency	Your handler should be idempotent — processing the same event twice should not produce duplicate records. Use the event id field to deduplicate.
Testing tool	Use ngrok (ngrok.com) to expose a local server for development and testing before registering a production endpoint.
Content-Type	Expect application/json (JWT payload). Do not assume plain JSON without decryption.

A5 — Registering Your Endpoint

Once your endpoint is built, secured, and tested, register it in Forms Logic:

5. Log in to your Forms Logic account as an administrator.
6. Navigate to Setup → Site Settings.
7. Locate the Webhook configuration section.
8. Enter your HTTPS endpoint URL.
9. Save. Forms Logic will begin sending events to your endpoint immediately.

> **Tip: Test before going live**
> Validate your endpoint using ngrok and a staging Forms Logic environment before registering your production URL. Confirm you can receive, decrypt, verify, and parse a test payload end-to-end.

A6 — Sample Full Payload (Decrypted)

The following is an example of the full decrypted JSON payload your endpoint will receive after a workflow_finalized event. Fields marked null may be populated depending on workflow configuration.


```
{
  "id": "456asas1MqqbKLt4dXK03v5qaIbiNCC",
  "event": "workflow_finalized",
  "apiVersion": "2024-04-01",
  "created": 16800645587,
  "type": "workflow",
  "data": {
    "id": "c0f87fab-3398-4def-9279-1df84288c98g",
    "organization": "The Organization Name",
    "activityType": "New-Account",
    "relationshipType": "Advisory (Fee)",
    "transactionType": "New Business Advisory/RIA Brokerage",
    "accountType": "Rollover IRA",
    "custodian": "Custodian",
    "state": "Finalized",
    "transactionAmount": "$60,000.00",
    "signatureType": "Docusign",
    "repCodeChoosen": "A1234",
    "advisor": {
      "id": 171095,
      "providerKey": "EmailOrIdFromExternalProvidedBySSO",
      "firstName": "firstName",
      "lastName": "lastName",
```

```
      "email": "email@organization.com",
      "repCode": "12345",
      "createdDate": "2024-07-06T11:58:21.344564Z"
    },
    "client": {
      "id": 207796,
      "providerKey": "EmailOrIdFromExternalProvidedByCRM",
      "clientType": "IndividualOrEntity",
      "firstName": "ClientName",
      "lastName": "ClientLastName",
      "email": "client@gmail.com",
      "dob": "1996-05-19",
      "home": { "address1": "1111 home address", "city": "City", "state": "ST",
"zip": "11111" },
      "createdDate": "2024-09-03T16:32:16.471747Z"
    },
    "forms": [
      { "id": "111111", "formName": "User Account", "isSigned": false,
        "quikFormId": "222222", "productId": "333333" }
    ],
    "docs": [
      { "id": "f05903fe-...", "fileName": "Revocable_Trust-
LAST_PRINT_REQUEST.pdf",
        "category": "SubmittedDocument", "formType": "Pdf" },
      { "id": "ea5fb512-...", "fileName": "Summary.pdf",
        "category": "AuditReport", "formType": "Pdf" },
      { "id": "608a03f5-...", "fileName":
"Revocable_Trust_Letter_of_Authorization.pdf",
        "category": "SignedDocument", "formType": "Pdf" }
    ],
    "customFields": [
      { "fieldName": "AnyFieldRequired", "fieldValue": "123" }
    ]
  }
}
```