iRaiser Fundraising for Salesforce

FAQ & Troubleshooting

iRaiser

Table of contents

Table of contents **2**

General **4**

What does the package do? 4

What is the supported API version and namespace? 4

Which Salesforce editions does it require? 4

Where is the source of truth for the architecture? 4

Installation and setup **4**

What's the minimum working configuration? 4

What webhook URL do I give iRaiser? 5

How do I know my Site guest user has the right permissions? 5

Configuration **5**

Which fields live on iRaiser_Setting__c? 5

Can I point the handler flow at a subscriber-local flow instead of the packaged template? 5

The template flow expects a specific input variable — what is it? 6

What does useApi control? 6

Supported object types **6**

Which iRaiser object types does the package accept? 6

How do I add support for a new object type? 6

How do I stop accepting a type I don't want? 6

Processing flow **7**

Why is there a platform event in the middle? 7

How do I know a record was processed? 7

Does the record-insert trigger fire on update? 7

Retry and recovery **7**

How do I retry a failed notification? 7

What options does the Retry modal expose? 7

Can I bulk-retry errored records? 7

What does the Retain state do? 7

Retention and cleanup **8**

How do I schedule the cleanup job? 8

What exactly gets deleted? 8

Can I change the retention window without a redeploy? 8

Does the batch chain itself? 8

Security and permissions **8**

Who needs which permission set? 8

Why does the Setup page say I'm not authorised? 9

Does the iRaiser API key touch the database? 9

Is the webhook endpoint authenticated? 9

Upgrades **9**

What happens to pending New and Error records during an upgrade? 9

Does upgrading reset my custom setting? 9

Will my handler flow break on upgrade? 9

Troubleshooting **9**
Webhook returns 400 "Invalid or no JSON body" 9
Webhook returns 500 with "An error occurred (...)" 9
Notification POST returned 200 but no record appears 10
Record stuck at State__c = 'New' 10
Record stuck at State__c = 'Error' 10
Every notification goes to Error immediately with "Callout not allowed" 10
Setup LWC shows a blank permission-set list 11
Retry modal says "Flow API Name is required" even though the default is populated 11
Cleanup batch never deletes anything 11
Apex tests fail locally with "You are not authorized to manage iRaiser setup" 11
"Duplicate value found: duplicates value on record" on PermissionSetAssignment 11
Deploys to the shared scratch org keep trying to delete SC_Retry_Webhook_Notification 11
How do I run LWC tests? 12
How do I cut a new package version? 12
How does namespace-prefix stripping work for flow names? 12
Why does NotificationService.cleanUpJson exist? 12
Where is the CRUD/FLS enforcement documented? 12
When you're still stuck **12**

This guide answers the questions administrators, developers and integrators most often ask about the package, followed by a troubleshooting playbook keyed on the symptom you see.

Related documents:

- **Admin Manual** — installation and step-by-step configuration.
- **Architecture** — the detailed end-to-end design of the package.

General

What does the package do?

It receives webhook notifications from the iRaiser fundraising platform, optionally enriches them with a full payload from the iRaiser REST API, and hands the enriched record to a Salesforce Flow that you own. Your flow does the mapping onto your data model (Accounts, Contacts, Opportunities and so on).

What is the supported API version and namespace?

API version 64.0 and namespace **COBRA**. Every field, object, class and flow shipped by the package is prefixed **COBRA__** once installed in a subscriber org.

Which Salesforce editions does it require?

Any edition that supports Apex, platform events and Sites. Most subscribers run Enterprise or above. Essentials and Professional without REST API access will not work, because webhooks need a Site with the REST resource enabled.

Where is the source of truth for the architecture?

The Architecture document. Where document and source tree disagree, the source tree wins.

Installation and setup

What's the minimum working configuration?

1. Install the managed package.
2. Configure the named credential **COBRA__Iraiser_NC** to point at your iRaiser API tenant.
3. Configure the external credential **COBRA__Iraiser_EC** with an API-key principal (iRaiser supplies the key).
4. Create a Salesforce Site and give its guest user the **COBRA__iRaiser_Webhook** permission set.
5. Assign **COBRA__iRaiser_Process_User** to the automated user that runs enrichment — typically the Automated Process user with alias autoproc.
6. Assign **COBRA__iRaiser_Admin** to your integration administrator.
7. Open the **iRaiser Setup** tab inside the iRaiser Lightning app and fill in the custom setting fields.
8. Save As the **COBRA__Template_iRaiser_Webhook_Handler** flow into your own namespace, customise the mapping, activate it, and point the setting at the new API name.
9. Schedule the cleanup job (see the retention section below).
10. Register the Site webhook URL with iRaiser.

What webhook URL do I give iRaiser?

```
https://<your-site-domain>/services/apexrest/COBRA/webhook
```

The Site's base URL is whatever you configured when creating the Site. No authentication header is required — the endpoint is public by design, protected by Site rate limits.

How do I know my Site guest user has the right permissions?

The guest user needs **Create** on **COBRA__Webhook_Notification__c** and access to the external credential principal. Assigning **COBRA__iRaiser_Webhook** covers both. You can verify with a test POST from your iRaiser environment, or with curl:

```
curl -X POST https://<site>/services/apexrest/COBRA/webhook \
  -H 'Content-Type: application/json' \
  -d '{"object_type":"donation","object_id":"12345",
       "event_type":"donations.update",
       "application":"Kentaa","site_id":"demo"}'
```

A 200 response means the record was accepted into **Webhook_Notification__c**.

Configuration

Which fields live on iRaiser_Setting__c?

Field	Default	Purpose
Use_Named_Credential__c	**COBRA__Iraiser_NC**	Named credential used for enrichment callouts
Notification_Handler_Flow__c	**COBRA__Template_iRaiser_Webhook_Handler**	Flow invoked per notification
Parameter_Name__c	**notification**	Name of the flow input variable
Retention_Days__c	**7**	Days to keep Processed notifications before cleanup

The org-default record is the one the runtime uses. Hierarchy overrides are available but rarely needed — the pipeline runs as the automated user, not per end user.

Can I point the handler flow at a subscriber-local flow instead of the packaged template?

Yes. Save As the template into your own namespace (or no namespace), activate it, then set **Notification_Handler_Flow__c** to the new API name — with or without the **COBRA__ prefix**. The runtime strips the prefix before invoking when it sees one, so both forms resolve.

The template flow expects a specific input variable — what is it?

A variable whose API name equals **Parameter_Name__c** (default **notification**), of Apex-defined type **COBRA.ExtendedNotification**, marked **Available for input**. If you rename the variable, update the custom setting to match; a mismatch makes the flow interview fail at start.

What does useApi control?

- **useApi = true** (default): the package makes a REST callout to iRaiser to fetch the full payload, deserialises it into the appropriate DTO, and hands the enriched **ExtendedNotification** to your flow.
- **useApi = false**: the package rehydrates the enriched payload from **Full_Payload__c** on the record, populated by a previous successful run. Useful for retries when the iRaiser API is down or the source record has been deleted upstream.

The toggle is only exposed on the Retry modal — incoming live webhooks always enrich through the API.

Supported object types

Which iRaiser object types does the package accept?

Whatever has a matching **COBRA__Supported_Notifications__mdt** record. The package ships seven: **action, company, donation, newsletter_subscription, project, segment** and **team**.

Any notification whose **object_type** doesn't match a record in this custom metadata is silently dropped at ingest. This is intentional — it lets you accept a firehose and filter server-side, without returning errors that iRaiser would retry against.

How do I add support for a new object type?

Four changes, all of them required:

1. Add a DTO class under **force-app/main/default/classes/** matching the iRaiser API shape (snake_case fields).
2. Add a new slot on **ExtendedNotification**.
3. Add a **when '<object_type>'** branch in **NotificationService.enrichByApi**.
4. Add a **COBRA__Supported_Notifications__mdt** record whose DeveloperName matches the lower-case **object_type**.

Skipping the custom metadata means the listener drops the notification before enrichment ever runs.

How do I stop accepting a type I don't want?

Delete or deactivate the corresponding **COBRA__Supported_Notifications__mdt** record. The listener starts dropping those notifications silently.

Processing flow

Why is there a platform event in the middle?

To decouple transactions. The REST listener inserts **Webhook_Notification__c** as the Site guest user — that transaction must be fast and cannot safely carry a callout. The Apex trigger publishes **iRaiser_Notification__e**, which is delivered in a fresh transaction with a fresh set of governor limits, as the automated user who owns the enrichment callout. Without the platform event hop, the guest user would need callout permissions and the transactional contract would be messier.

How do I know a record was processed?

State__c transitions from New to either Processed or Error. **Last_Processed__c** carries the timestamp of the last attempt, and **Last_Run_Messages__c** holds either the invoked flow name (on success) or the exception message (on error).

Does the record-insert trigger fire on update?

No. It fires on **after insert** only, and filters to **State__c = 'New'**. Manually resetting the state back to New does not re-trigger processing — use the Retry action instead (see below).

Retry and recovery

How do I retry a failed notification?

Open the **Webhook_Notification__c record**, click the **Retry** quick action, pick your options in the modal and confirm. The modal is a Lightning Web Component (**webhookNotificationRetry**) that calls **WebhookNotificationRetryController.retry**, which wraps the record in a **NotificationProcessor** and hands it to the same invocable the platform event subscriber uses.

What options does the Retry modal expose?

- **Flow API Name** — defaults to the effective **iRaiser_Setting__c** value; override it to test a different flow.
- **Parameter Name** — defaults to **Parameter_Name__c**, overridable.
- **Use iRaiser API** — when off, the retry uses **Full_Payload__c** from the record; when on, it makes a fresh callout.

Can I bulk-retry errored records?

Not through the UI. The retry controller enforces one record per invocation, mirroring **NotificationProcessor.processWebhookNotifications**. For bulk recovery, write a one-off Apex script or an invocable-driven flow and iterate — each invocation schedules its own **@future**.

What does the Retain state do?

It's the opt-in escape hatch from cleanup. Records with **State__c = 'Retain'** are never deleted by **WebhookNotificationCleanupBatch**, regardless of **Retention_Days__c**. Use it for anything you need to preserve for audit, debugging or compliance.

Retention and cleanup

How do I schedule the cleanup job?

Open the Developer Console → **Debug** → **Open Execute Anonymous** and run:

```
System.schedule(
  'Webhook Notification Cleanup',
  '0 0 2 * * ?',
  new COBRA.WebhookNotificationCleanupScheduler());
```

The cron expression above runs daily at 02:00. Tune it to your traffic — a nightly run is enough for most volumes; for extremely chatty integrations you may want every six hours.

What exactly gets deleted?

Webhook_Notification__c records where:

- **State__c = 'Processed'**, and
- **Last_Processed__c** is older than now minus **Retention_Days__c** days.

Records in New, Error or Retain are never deleted.

Can I change the retention window without a redeploy?

Yes — edit **Retention_Days__c** on the org-default iRaiser_Setting__c through the Setup LWC. The next cleanup run picks it up.

Does the batch chain itself?

Yes — up to 10,000 records per chunk. If more remain, **finish()** schedules another batch (suppressed during **Test.isRunningTest()** to avoid test recursion).

Security and permissions

Who needs which permission set?

Permission set	Audience
COBRA__iRaiser_Webhook	Site guest user receiving webhooks. Create-only on **Webhook_Notification__c**; external credential principal access.
COBRA__iRaiser_Process_User	Automated user (typically autoproc) running enrichment. Edit on **Webhook_Notification__c**; access to the enrichment classes.
COBRA__iRaiser_Admin	Administrators. Full CRUD and Modify All; tab visibility; required to use the Setup LWC.

Why does the Setup page say I'm not authorised?

iRaiserSetupController gates every permission-management endpoint behind an assertIsIraiserAdmin() check that looks for a **COBRA__iRaiser_Admin** assignment. Assign the permission set to yourself and refresh.

Does the iRaiser API key touch the database?

No. It lives on the external credential principal and is surfaced to callouts through the named credential. No Apex reads it and no custom field stores it.

Is the webhook endpoint authenticated?

No — it's a public REST resource on a Salesforce Site. iRaiser signs nothing, so Salesforce trusts the source by virtue of the URL being unguessable (it's yours) and rate-limits the Site. If that is not acceptable for your threat model, wrap the Site with a reverse proxy that enforces a shared-secret header and verify it inside **WebhookListener** before delegating — no such check ships by default.

Upgrades

What happens to pending New and Error records during an upgrade?

They remain in the object. Trigger behaviour on the next insert is unchanged, and existing Error records can still be retried. Don't flip the state back to New hoping the trigger re-fires — it won't.

Does upgrading reset my custom setting?

No. **iRaiser_Setting__c** is customer-owned data; upgrades do not touch it.

Will my handler flow break on upgrade?

Only if the structure of **COBRA.ExtendedNotification** changes in a way your flow relies on. Breaking DTO changes are avoided on minor versions. Check the release notes in the **packaging-commands.txt** commits for the specifics of each version.

Troubleshooting

Webhook returns 400 "Invalid or no JSON body"

iRaiser sent an empty body — most often with misconfigured test tooling. Check what iRaiser is actually POSTing: the body must be a JSON object with at least **object_type**, **object_id** and **event_type**.

Webhook returns 500 with "An error occurred (...)"

The JSON deserialised, but NotificationService.storeIncomingNotification threw. The message tail in the response is the root cause. Common culprits:

- The guest user lacks **Create** on **COBRA__Webhook_Notification__c**. Fix: assign **COBRA__iRaiser_Webhook**.

- The guest user lacks read access to **COBRA__Supported_Notifications__mdt**. The permission set grants this — re-check the assignment.
- A required field on **Webhook_Notification__c** is missing from the incoming body. Validate against the **Notification** DTO shape.

Check **Setup → Debug Logs** for the Site guest user for the full stack trace.

Notification POST returned 200 but no record appears

The **object_type** isn't in **COBRA__Supported_Notifications__mdt**. By design, unknown types are silently dropped with a 200 response. Add the metadata record — and a matching DTO branch if you intend to enrich it — and retry.

Record stuck at State__c = 'New'

The platform event chain isn't firing or isn't being subscribed to. Check in order:

1. Is **WebhookNotificationTrigger** active? **Setup → Apex Triggers**.
2. Is **iRaiserNotificationTrigger** active?
3. Look at the automated user's debug logs for exceptions at the platform event subscriber — a failing subscriber keeps the record on New.
4. Is the automated user assigned **COBRA__iRaiser_Process_User**?

Record stuck at State__c = 'Error'

Read **Last_Run_Messages__c** — it holds the exception. Common patterns:

- **"Fetching failed 401" →** the API-key principal is misconfigured on the external credential. Reset the secret in **Setup → Named Credentials**.
- **"Fetching failed 404" →** the iRaiser-side record was deleted. Retry with **useApi = false** to use the stored payload.
- **"Unexpected response:" →** the iRaiser API returned an object in an unexpected shape, not wrapped with the expected top-level key. Inspect **Last_Run_Messages__c** for the raw body.
- **"Flow does not exist" →** **Notification_Handler_Flow__c** points at a flow that is inactive or missing. Verify the flow API name and its activation state.
- **"No matching input variable" →** **Parameter_Name__c** doesn't match a flow input variable, or the variable's type isn't **COBRA.ExtendedNotification**, or it isn't marked Available for input.

Every notification goes to Error immediately with "Callout not allowed"

processNotification is **@future(callout=true)**. If a caller invokes it outside the future context — for example from a trigger in a synchronous path — the callout permission is lost. This shouldn't happen with the shipped wiring, but custom modifications can break it. Check that every path into **NotificationService.processNotification** goes through the invocable/trigger pipeline or a **@future** boundary.

Setup LWC shows a blank permission-set list

Either the running user lacks **COBRA__iRaiser_Admin** (the controller gate returns an error that the LWC swallows), or the **autoproc** user doesn't exist in your org. Assign the admin permission set and verify that **SELECT Id FROM User WHERE Alias = 'autoproc'** returns a row.

Retry modal says "Flow API Name is required" even though the default is populated

The hierarchy custom setting returned null because no org-default record exists yet. Save the Setup tab once — saving upserts the org default — then reopen the Retry modal.

Cleanup batch never deletes anything

1. Is the scheduler actually scheduled? **Setup → Scheduled Jobs**.
2. Are the records old enough? **Last_Processed__c** must be older than now minus **Retention_Days__c**.
3. Are the target records in Processed? New, Error and Retain are immune.

If the job runs but nothing is deleted, verify the query by hand:

```
SELECT COUNT() FROM COBRA__Webhook_Notification__c
WHERE COBRA__State__c = 'Processed'
AND COBRA__Last_Processed__c < :DateTime.now().addDays(-7);
```

Apex tests fail locally with "You are not authorized to manage iRaiser setup"

The running user isn't assigned to **iRaiser_Admin**. The test class has a **@TestSetup** that assigns it idempotently, but if your org is missing the **COBRA__iRaiser_Admin** permission set (unusual — it ships with the package) that setup is a no-op. Assign it manually and re-run.

"Duplicate value found: duplicates value on record" on PermissionSetAssignment

The admin toggled a permission set that was already assigned and the stale LWC cache hadn't refreshed. Refresh the Setup tab — **getPermissionStatus** is **cacheable=true**, so the UI state can lag reality briefly.

Deploys to the shared scratch org keep trying to delete SC_Retry_Webhook_Notification

Legacy source-tracking state. Run:

```
sf project reset tracking --target-org <alias> --no-prompt
```

For a single method:

```
sf apex run test --tests NotificationServiceTest.testEnrichDonation \
  --result-format human --wait 10 --target-org <alias>
```

How do I run LWC tests?

```
npm run test:unit            # one-shot
npm run test:unit:watch      # interactive
npm run test:unit:coverage   # coverage report
```

How do I cut a new package version?

```
sf package version create \
  --package "iRaiser Fundraising for Salesforce" \
  --code-coverage --installation-key-bypass \
  --target-dev-hub cobra --wait 10
```

The full release history and the promotion commands live in **packaging-commands.txt**.

How does namespace-prefix stripping work for flow names?

NotificationService.startFlow checks whether the configured flow name starts with **cobra__** (case-insensitive). If so, it strips the prefix and passes **COBRA** as the namespace argument to **Flow.Interview.createInterview**. If not, both namespace and flow name are passed as-is. This lets the same configuration value work for the packaged template and for subscriber-local flows.

Why does NotificationService.cleanUpJson exist?

iRaiser's API returns fields named **currency** in several places. **currency** is a reserved Apex identifier and cannot be used as a DTO property name, so the service rewrites the JSON literal **currency** to **currencyCode** before deserialisation. If you add a DTO whose field name collides with another Apex reserved word, extend **cleanUpJson** rather than renaming the API field.

Where is the CRUD/FLS enforcement documented?

In the Architecture document, chapter 6. In short: SOQL uses **WITH USER_MODE**, DML uses **AccessLevel.USER_MODE**, and write-back in the @future method uses **Security.stripInaccessible(AccessType.UPDATABLE, ...)** before the update.

When you're still stuck

- Send a note with the record Id, the State__c, the tail of Last_Run_Messages__c and the timestamp of the failure.
- Include the relevant debug log excerpt from the automated user for that timestamp if possible.
- For webhook-side failures, include the HTTP response body iRaiser recorded for the POST.

That combination is usually enough to find the root cause without a round trip.

Contact support@cobracrm.nl for more support.