# Integration patterns: push and pull, batch and real time

The basic integration patterns applied to Exponential - who starts the exchange (push or pull), how often (batch or near real time), delta and full runs, outbox and inbox, polling with a watermark, and which Exponential...

Source file: `doc/eng-us/help/integrators/integration-patterns-push-pull-batch-real-time.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/integrators/integration-patterns-push-pull-batch-real-time.md

This page is for architects and developers who design data flows between Exponential and other systems. It names the basic patterns, says when each fits, and maps each to the surface that implements it in Exponential. The examples assume a product feed from an ERP and a lead flow to a CRM, the two most common jobs.

## Contents

1. Two questions per flow
2. The four combinations
3. Delta runs and watermarks
4. Outbox and inbox
5. Reconciliation
6. Choosing per flow

## Two questions per flow

1. **Who starts the exchange?** In a **push**, the system that has the change sends it. In a **pull**, the system that needs the change asks for it.
2. **When?** In a **batch**, changes are collected and moved together on a schedule. In **near real time**, each change moves within seconds or minutes of happening.

## The four combinations

| | Into Exponential | Out of Exponential |
|---|---|---|
| **Push, near real time** | The other system calls the REST create route or `expservices` writes for each change | A workflow event on `content / publish / after` records the change; a frequent cronjob sends it (outbox). The audit webhook sink for event feeds |
| **Push, batch** | The other system drops a file (SFTP, a folder); a cronjob part imports it; or `xrowextract` schedules an import | A cronjob part or an `xrowextract` schedule writes a file and delivers it to SFTP, S3, WebDAV or an HTTP endpoint |
| **Pull, near real time** | A frequent cronjob part asks the other system's API for changes since the last watermark | The other system polls the REST API or `expservices` (recently modified objects), or an RSS export |

| | Into Exponential | Out of Exponential |
|---|---|---|
| **Pull, batch** | A nightly cronjob part downloads the full export of the other system | The other system downloads a nightly export or a feed |

Push from Exponential needs code on the Exponential side; pull by the other system needs only credentials. Pull into Exponential needs the other system's API and a cronjob part.

## Delta runs and watermarks

A **full run** moves everything every time: simple, slow, and it finds removals by absence. A **delta run** moves only what changed since the last successful run, identified by a **watermark**:

| Watermark | Where it comes from in Exponential |
|---|---|
| Modification time | `eZContentObject` `modified`; the `expservices` service `expobject::recentlyModified`; `xrowextract --changed-since` |
| An increasing id | `ezinfocollection.id` for form answers; your outbox table's id |
| The other system's change token | Stored by your integration after each run |

Rules: store the watermark only after the run succeeded; overlap the window a little (for example five minutes) and rely on idempotency to absorb duplicates; run a full reconciliation now and then to catch what deltas miss, such as removals.

## Outbox and inbox

| Pattern | What it is | In Exponential |
|---|---|---|
| **Outbox** | The change is recorded in a table inside the same system and transaction; a separate process sends it and marks it sent | A workflow event or event listener writes a row to your extension's table; a cronjob part or an expqueue worker sends it |
| **Inbox** | Incoming messages are stored first, then processed; duplicates are recognized by message id | A module view or server function stores the payload in your table and answers at once; a cronjob part processes it |

Both decouple the editor's request from the other system's availability, and both make retries safe.

## Reconciliation

Even well-built syncs drift: a message lost while a system was down, a record changed by hand. Plan a **reconciliation** job that compares both sides by key (the `remote_id`) and reports or repairs differences:

- ☐ Count of records per class on each side.
- ☐ Records present on one side only (created or removed outside the sync).
- ☐ A checksum or modification time per record to find changed ones.
- ☐ A report to the operators, and a decision rule for who wins.

## Choosing per flow

| Flow | Typical choice |
|---|---|
| ERP products into product pages | Push batch (file) or pull batch (API) nightly, with a delta during the day |
| Price and stock | Pull near real time, or a server function reading a cache your import keeps |
| Leads from forms to the CRM | Pull near real time by a cronjob from Exponential's side, pushing to the CRM API |
| Publications to a search or archive system | Outbox, near real time |
| Editorial content to a mobile app | Pull by the app over REST, cached on the app side |
| Users from a directory | Login handler (on demand) plus a nightly de-provisioning job |

## Related pages

- Choosing an integration surface
- Idempotent imports keyed by remote_id
- Workflows and triggers as integration hooks
- Background work patterns
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.