

1

CHAPTER 1

The Production Reality Check

Intermediate

Edition 1.0

Getting a message into CargoWise is a weekend project. Running eAdapter as the pipe your business depends on is a different discipline entirely, and almost none of it is written down in one place. This chapter is the operational model: what the eAdapter actually is, when you should be using something else, and the five failure modes waiting in your first year.

IN THIS CHAPTER

- From “it works” to “it runs the business”
- The eAdapter operational model in five constraints
- Async everything: where status actually lives
- eHub vs E2E vs eAdapter: the decision framework
- The knowledge barrier, and the tools that bridge it
- The five failure modes of year one

From “it works” to “it runs the business”

Every eAdapter integration has the same honeymoon. You get credentials, you push your first XML, a job appears in CargoWise, and someone says “ship it.” The demo works. The pilot works. Then the integration stops being a project and becomes **infrastructure**, the thing that creates every job, files every milestone, and moves every invoice, and the rules change underneath you.

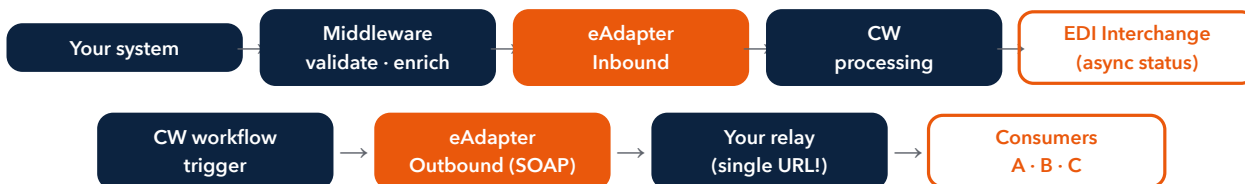
In production, nobody asks whether a message **can** get into CW. They ask why Tuesday's bookings are missing, which of the four thousand messages this month silently failed, and who is getting paged about it. The gap between those two worlds is what this volume covers. None of it requires genius; all of it requires knowing where the sharp edges are **before** your business is resting on them.

The industry's own diagnosis agrees. As one analysis of CargoWise integration problems puts it: most problems come from a lack of setup, training, or customisation, not from flaws in the software. The eAdapter is capable. It is also, in the words of a G2 reviewer, "very good if you have your own developers", a compliment with a warning folded inside it.

The operational model in five constraints

Strip away the XML schemas and the eAdapter's production behaviour reduces to five architectural constraints. Everything in this book is a consequence of one of them:

1. **Single outbound endpoint**, requires a relay/proxy for multi-system integration.
2. **Async processing**, no immediate success/failure response on send.
3. **XML-only**, requires transformation for non-XML consumers.
4. **UI defaults don't apply**, API messages must be self-contained.
5. **Field locking after customs commencement**.



The whole operational surface: one inbound path whose status you must go looking for, one outbound pipe that fans out only if you build the fan.

Async everything: where status actually lives

The constraint that reshapes your architecture most is the second one. Send a message inbound and CargoWise hands you a **tracking UUID**, not a result. As the Steele Consulting library documentation puts it, the send call "doesn't return message status directly"; you check the **CW1 EDI Interchange** to find out what actually happened to your message. Success, rejection, or silent mangling, the answer lives inside CargoWise, minutes later, and your integration has to go and get it.

That single fact generates most of Volume 2: if send is fire-and-wait, you need message-status visibility (Chapter 3), you need to know why an expected outbound message never fired (Chapter 4), and you need timeouts and partial-failure handling for everything (Chapter 7).

WARNING

One protocol note before anything else. The community knowledge base describes the inbound service as a **RESTful XML interface**, and CargoMode’s production integration talks to it exactly that way, plain HTTP+XML: raw XML body, basic auth, no envelope, no GZIP. The open-source client libraries, meanwhile, wrap their sends in a SOAP/WSDL envelope (`eAdapterStreamedService.wsdl`) that abstracts the compression and wrapping for you. Both paths are documented realities, verify which one **your** instance and tooling expect before writing a line of code. The outbound service, in every source that documents it, is a SOAP push to your configured URL.

Inbound accepts three XML dialects, Universal XML (the default), Native XML, and legacy Standard XML, through a single RESTful XML endpoint per CW instance, authenticated with credentials configured in the CW Registry. Outbound emits XUS (full shipment data) and XUE (event notification) messages, and **only** when a Completion Trigger Action is attached to the workflow milestone, a detail that quietly deletes messages you assumed were automatic (Chapter 4).

Choosing your weapon: eHub, E2E, or eAdapter

Here is the least-told truth in this book: **the best eAdapter integration is often the one you don’t build**. CargoWise offers three main integration paths, and eAdapter is the most powerful and the most expensive to own:

- **eHub**, pre-built carrier connections for ocean carriers, air freight carriers, and NVOCCs.
- **E2E Messaging**, standardised message exchange over CW’s global network for standard EDI formats (X12 850, 856, 315, and so on), between CW-to-CW users.
- **eAdapter**, custom integrations with full flexibility; requires development resources.

The community-documented decision framework:

Scenario	Recommended option
Two CW-to-CW users	E2E Messaging
Connecting to a major carrier	eHub (pre-built)
Custom integration with ERP/WMS	eAdapter
Partner uses standard EDI (X12, EDIFACT)	eAdapter with EDI transform
Need multiple outbound destinations	eAdapter + middleware relay

The default order of preference is **eHub over E2E over eAdapter**, reach for eAdapter only when the simpler options can’t meet the need. Pre-built connections are more reliable and require less maintenance than anything custom. And whichever you choose, choose **deliberately**: mixing patterns without a strategy creates duplicated flows, inconsistent data, and confusion. The sourced discipline is

an integration strategy document, map every trading partner and system to its EDI option, and maintain a registry that prevents duplicate flows.

CRITICAL

If you do land on eAdapter, do not underestimate the middleware requirement, budget for it as a core infrastructure component, not an afterthought. CW users are told they **must** host their own outbound service and middleware, to manage queues, retries, transforms, and monitoring, and many smaller forwarders underestimate this, ending up with fragile scripts or ad-hoc services that break in production. Chapters 2 and 3 are the blueprint for doing it properly.

The knowledge barrier

There is a structural reason you’re reading a third-party book about this. The eAdapter Developer Guide isn’t made available until a CargoWise customer purchases the eAdaptor modules, a knowledge barrier that means you commit before you can study. Public community knowledge is thin: Reddit content on CargoWise integration is extremely sparse (the logistics, supply-chain, and freight subreddits carry almost no CW-specific technical discussion), and what exists lives scattered across G2 reviews, LinkedIn posts, consultant blogs, and the WiseTech partner ecosystem. G2 reviewers add that CW’s customer service runs below expectations for resolving integration issues, so when something breaks, the answer usually has to come from your side of the fence.

Three open-source reference implementations bridge the gap, letting your team learn the message mechanics before the paywalled guide arrives:

Tool	Type	Language	Licence
Steele Consulting eAdapter	Client library	Node.js/TS	MIT
yob/cargowise	SOAP wrapper	Ruby	Open source
oliverbj/cord	Client library	PHP	Open source

The sourced recommendation that goes with them: invest in building some internal expertise even if you use consultants. Having at least one person who understands eAdapter basics dramatically reduces your dependency and your response time.

The five failure modes of year one

Every long-running eAdapter integration meets the same five walls. This book orders them the way you’re likely to hit them:

#	The wall	Chapter
1	The single outbound URL. Your second consumer arrives and CW has nowhere to send it, the relay you should have built on day one.	Ch. 2
2	Where did my message go? Async sends, tracking UUIDs, and building the status visibility CW won't push to you.	Ch. 3
3	The milestone that never fired. Workflows complete, and without a Completion Trigger Action, no message ever leaves.	Ch. 4
4	Field locking. Customs commences and fields you update every day stop accepting updates.	Ch. 5
5	The defaults gap. API-created jobs skip the org- and port-based defaults the UI applies, and nobody tells you.	Ch. 6

Chapters 7 and 8 close the loop with the disciplines that keep all five survivable: timeout budgets, splitting, and retry hygiene at the edge (Ch. 7), and the test environments, fixtures, and replay harness that turn every production failure into a permanent regression test (Ch. 8).

None of these walls is exotic. Every one of them is documented, in consultant blogs, in G2 reviews, in the scars of teams who run this in production. What this volume does is put them in one place, in the order you'll need them, with the fixes attached. Welcome to production.

Vol. 2. Chapter 1. Sources: the CargoWise eAdapter knowledge base (integration options, operational constraints, community insights via G2/consultant documentation), and CargoMode's production integration field notes.

Get the complete guide and trigger-map worksheet at cargomo.de.