FIELD MANUAL

The EWM Technical Architect's Manual

End-to-End Technical Reference for SAP Extended Warehouse Management

The EWM data model and process objects. Warehouse tasks, orders and storage control. The extension hierarchy, BAdI landscape and PPF. The RF framework and MFS. Decentralized integration design. Performance, testing and the EWM diagnostic sequence. Two requirements worked end to end, and a staged path to the architect role.

Prepared for Sayan Samanta
SAP Technology Lead
August 2026

Contents

How to Use This Manual

This is written for an experienced ABAP developer moving into EWM as a technical architect. It assumes you know ABAP, CDS, RAP and OO design, and teaches the EWM-specific layer: the data model, the process objects, the enhancement framework, RF, MFS, integration and the architecture decisions.

Read this warning before you read anything else.

EWM object names — BAdI names, function modules, table names, class names, customizing paths — are **release-specific and change between EWM 9.x, embedded S/4HANA EWM, and Decentralized EWM**. Everything in this manual that names a repository object is marked as one of:

- **[VERIFIED-STABLE]** — long-standing and unlikely to differ
- **[VERIFY]** — the concept is correct, the exact name must be confirmed in your target system before use

Nothing here should be copy-pasted into a client system without checking the object exists in that release. The code samples are structural — they show the shape of the solution, the sequencing, and the error handling. Treat them as design, not as production source.

How to verify, in order: 1. ADT → Released Objects, and the object's API State tab 2. `SE18` / enhancement spot browser for BAdIs in the `/SCWM/` namespace 3. The SAP EWM development and extension guide for your release 4. Where-used on a standard EWM program that does the same thing

The structure, the decision framework and the architecture content do not have this problem. Those are stable.

The ten parts

- **Part I — Foundations.** Landscape choices, the data model, master data.
- **Part II — Process Objects.** Delivery, warehouse task and order, storage control, handling units, stock.
- **Part IV — The Extension Framework.** Where EWM lets you hook in, and the discipline for choosing which hook.
- **Part V — RF and MFS.** The two areas that make an EWM developer scarce.
- **Part VI — Integration.** ERP↔EWM, external systems, and the decentralized question.
- **Part VII — Quality.** Performance, testing, debugging in EWM.
- **Part VIII — The Architect's Deliverables.** Warehouse structure design, volumetrics and NFRs, the authorisation model, UI and analytics strategy, business continuity, cutover and stock migration, and the WM→EWM and EWM 9.x→S/4 migration paths.
- **Part IX — Worked Examples.** Two requirements end to end.

- **Part X — The Architect Path.** What to learn, in what order, and how to get the delivery experience that actually makes you one.

Part I — Foundations

Chapter 1: EWM in the Landscape

1.1 The deployment options

You must be able to draw this and explain the trade-offs from memory. It is the first question in every EWM architecture conversation.

	Embedded EWM	Decentralized EWM	SCM EWM 9.x
Runs in	The S/4HANA system	Its own S/4HANA system	Its own SCM system
Integration to ERP	Internal, no queue	qRFC / core interface	qRFC / core interface
Upgrade coupling	Coupled to S/4	Independent	Independent
Master data	Shared	Distributed via CIF	Distributed via CIF
Best for	Simpler warehouses, single-system landscapes	High-volume, high-availability, multi-ERP	Legacy — end of maintenance

The architectural trade-off, stated honestly:

Embedded removes the interface layer entirely — no distribution, no queue monitoring, no master data replication. That is a large operational simplification and it is why SAP positions it as the default.

Decentralized buys you **independence**. The warehouse keeps running when ERP is down or upgrading. For a 24/7 distribution centre where an hour of downtime is measured in trucks, that is not a preference, it is a requirement. It also lets one warehouse serve multiple ERP systems.

The question that decides it: *"What happens to the warehouse when the ERP system is unavailable for four hours?"* If the answer is "it stops and that costs us serious money," you are decentralized. Everything else is secondary.

1.2 The migration context — why EWM technical is scarce right now

Two forces create the current demand:

- Legacy WM compatibility in S/4HANA has reached its end, so WM customers must move.
- SCM EWM 9.x reaches end of maintenance, so existing EWM customers must move to embedded or decentralized S/4HANA EWM.

Both migrations are **technical**, both land in heavily customized industrial landscapes, and both need someone who can read the old custom code and rebuild it in the new framework. That is the niche. It is spread across a long window rather than compressed, which makes it a durable specialisation rather than a spike.

1.3 What makes EWM development different from ordinary ABAP

Be clear about this early, because it is why the skill is scarce:

1. **Almost nothing is a normal transaction.** Warehouse execution runs through the RF framework, which has its own screen model, its own transaction concept, and its own customizing. Ordinary Dynpro and Fiori skills do not transfer.
2. **The document model is generic.** Deliveries use a generic document layer (`/SCDL/`) rather than the ERP delivery structures. You cannot reason from LIKP/LIPS habits.
3. **Process control is customizing-driven.** Storage control, warehouse process types and warehouse order creation rules mean behaviour that looks like code is often configuration. **The first debugging skill in EWM is knowing whether you are looking at a code problem or a customizing problem.**
4. **Real-time physical coupling.** In an automated warehouse, MFS talks to PLCs. A defect stops a conveyor. The testing and error-handling bar is higher than in ordinary business ABAP.
5. **Enhancement is BAdI-heavy.** EWM ships a large, well-structured BAdI landscape. Knowing which one fires when is most of the job.

Chapter 2: The Data Model

You cannot debug EWM without knowing where data lives. This chapter is the map.

2.1 The core object groups

Group	Holds	Namespace
Warehouse structure	Warehouse number, storage type, section, bin, activity area	`/SCWM/`
Stock	Quants, physical stock, available quantity	`/SCWM/`
Handling units	HU header, HU items, nested HUs	`/SCWM/`
Warehouse task / order	Open, confirmed and historical tasks; warehouse orders	`/SCWM/`
Delivery	Generic document header, items, statuses, references	`/SCDL/`
Master data	Warehouse product, packaging specification, resource	`/SCWM/` and `/SAPAPO/`

2.2 Tables worth knowing by heart

[VERIFIED-STABLE] — these are long-standing:

Table	Contents
`/SCWM/T300`	Warehouse numbers
`/SCWM/T301`	Storage types
`/SCWM/LAGP`	Storage bins
`/SCWM/AQUA`	Available quantity — the stock table you will use most
`/SCWM/HUHDR`	Handling unit header
`/SCWM/HUITM`	Handling unit items
`/SCWM/ORDIM_O`	**O**pen warehouse tasks
`/SCWM/ORDIM_C`	**C**onfirmed warehouse tasks
`/SCWM/WHO`	Warehouse orders

The single most useful thing in this chapter: the `ORDIM_O` / `ORDIM_C` split. An open task and a confirmed task live in **different tables**. Custom code that reads only `ORDIM_O` and reports "task not found" after confirmation is a defect you will see repeatedly, and it is the first thing to check when someone reports a task disappearing.

[VERIFY] — concept certain, name to confirm in your release: warehouse process type customizing, storage control tables, warehouse order creation rule tables, activity area assignment, RF customizing tables. Find them via the IMG node and then `SE11`, rather than trusting a name from memory or from a model.

2.3 The delivery document model

EWM does **not** use the ERP delivery tables. It uses a generic document model in the `/SCDL/` namespace with a consistent header / item / status / reference structure shared by inbound and outbound.

Practical consequences:

- Read deliveries through the delivery management API, never by selecting from tables directly. The model is normalised in ways that make direct reads fragile and wrong.
- Status is a separate object, not a field on the header. "Is this delivery complete?" is a status query, not a header read.
- The mapping between the ERP delivery and the EWM delivery is maintained by the distribution layer and is not a simple key equality.

2.4 How to explore the model in a real system

The method matters more than any table list, because the list goes stale:

1. Run the process in the UI, note the document numbers.
2. Use the display transactions to see the object structure.
3. Turn on an SQL trace over one process step and read which tables were actually touched. **This is the fastest way to learn EWM's real data model** and it works on any release.

4. Where-used on the tables you find, to locate the standard code that maintains them.
5. Read that standard code. EWM's own implementation is the best available documentation of its extension points.

Do step 3 on your first day with system access. Ninety minutes of SQL trace over a putaway and a picking cycle teaches more than a week of reading.

Chapter 3: Master Data

3.1 The objects

Object	Purpose	Technical note
Warehouse product	Material master data specific to a warehouse and party entitled to dispose	Distinct from the ERP material master; warehouse-specific fields drive putaway and removal
Storage bin	Physical location	Carries bin type, section, aisle/stack/level coordinates used by sorting
Packaging specification	How a product is packed, in levels	Drives HU creation, palletisation, and warehouse task quantity splitting
Resource	A person or device executing work	Bound to an RF user session; drives queue assignment
Handling unit	A physical unit of stock with an identifier	Nestable; the central object in most warehouse processes

3.2 Packaging specifications — the one to understand deeply

Packaging specifications are the most under-appreciated technical object in EWM. They drive:

- How much stock goes on one pallet, which determines warehouse task splitting
- Automatic HU creation during putaway and picking
- Palletisation in production and kitting scenarios
- Label content and count

A large share of "why did EWM create three tasks instead of one" questions resolve to packaging specification determination, **not** to warehouse order creation rules where people usually look first. Learn the determination logic before you learn WOCR.

3.3 Master data in decentralized landscapes

In a decentralized deployment, material, customer, vendor and batch data are replicated from ERP. Technical consequences you will own as architect:

- Replication failures manifest as process errors deep in warehouse execution, not as obvious interface errors. Monitoring must cover replication, not just the delivery queue.
- Custom fields on the ERP material master do **not** arrive in EWM automatically. Extending replication is a specific piece of work and it is routinely forgotten until UAT.
- Timing: a delivery can arrive referencing a product that has not replicated yet. Design for it — it is a normal condition, not an exception.

Part II — Process Objects

Chapter 4: Deliveries

4.1 The lifecycle

Inbound: ERP purchase order → ERP inbound delivery → distributed to EWM → EWM inbound delivery notification → EWM inbound delivery → unloading → goods receipt → putaway warehouse tasks → confirmation.

Outbound: ERP sales order → ERP outbound delivery → distributed to EWM → EWM outbound delivery order → wave → pick warehouse tasks → packing → staging → loading → goods issue → confirmation back to ERP.

Note the asymmetry: inbound has a *notification* stage before the actual delivery; outbound has a *delivery order* which is EWM's planning document and does not map one-to-one to the ERP delivery. Architects who assume symmetry get the integration design wrong.

4.2 Reading a delivery in code

Always through the delivery management API, never direct table selects.

```
" [VERIFY] class and method names against your release
DATA(lo_dlv) = /scwm/cl_dlv_management_prd=>get_instance( ).

DATA(ls_read_options) = VALUE /scwm/dlv_query_contr_str(
  data_retrival_only = abap_true
  head_deep          = abap_true
  item_deep          = abap_true ).

DATA(ls_include_data) = VALUE /scwm/dlv_query_incl_str_prd(
  head_status  = abap_true
  head_partyloc = abap_true
  item_status  = abap_true ).

TRY.
    lo_dlv->query(
      EXPORTING
        iv_doccat      = wmegc_doccat_pdo      " outbound delivery order
        it_docid       = lt_docid
        is_read_options = ls_read_options
        is_include_data = ls_include_data
      IMPORTING
        et_headers     = DATA(lt_head)
        et_items       = DATA(lt_item) ).
  CATCH /scdl/cx_delivery INTO DATA(lx_dlv).
    " handle — never ignore
ENDTRY.
```

Design points:

- `head_deep` and `item_deep` control how much is read. Reading everything for every delivery is a performance defect — request only what you need.

- Document category (`doccat`) distinguishes inbound delivery, inbound notification, outbound delivery order and outbound delivery. Getting it wrong returns nothing and looks like missing data.
- Status is in the status structure, not on the header.

4.3 Delivery distribution (decentralized)

The ERP delivery is distributed to EWM and confirmations flow back. Technically this is queued communication, and the failure modes are queue failure modes:

Failure	Symptom	Where to look
Queue blocked	Deliveries stop arriving, no error in EWM	qRFC monitor, inbound queue
Mapping error	Delivery arrives with missing or wrong data	Distribution log, mapping BAdI
Master data missing	Delivery rejected in EWM	Replication status for the product
Confirmation not returning	ERP shows delivery open after EWM goods issue	Outbound queue

Architect's rule: a decentralized EWM design is not complete without a queue monitoring and alerting design. This is the most common gap in EWM implementations and the most common cause of "the warehouse is fine but ERP thinks nothing shipped."

Chapter 5: Warehouse Task and Warehouse Order

The central execution objects. Everything a worker does is a warehouse task, grouped into a warehouse order.

5.1 The concepts

- **Warehouse Task (WT)** — one movement instruction: move this quantity of this product from this source to this destination. Created, then confirmed or cancelled.
- **Warehouse Order (WO)** — the work package assigned to a resource. Contains one or more WTs. Created by **Warehouse Order Creation Rules (WOCR)**.
- **Warehouse Process Type (WPT)** — the configuration object that determines how a task behaves: which stock types, which storage control, which confirmation requirements.

The WPT is to EWM what a movement type is to classic MM, but with far more behaviour attached. Most "why did it do that" questions terminate at the WPT.

5.2 Creating a warehouse task

```
" [VERIFY] parameter names against your release
DATA: lt_create TYPE /scwm/tt_to_crea_prod,
      lt_ordim  TYPE /scwm/tt_ordim_confirm.

lt_create = VALUE #( ( matid    = lv_matid
                       quan     = lv_quantity
                       unit     = lv_uom
                       procty   = lv_process_type
                       vlpla    = lv_source_bin
                       nlpla    = lv_dest_bin
                       entitled = lv_entitled ) ).

CALL FUNCTION '/SCWM/TO_CREATE'
  EXPORTING
    iv_lgnum      = lv_warehouse
    iv_commit_work = abap_false          " control the LUW yourself
    it_create     = lt_create
  IMPORTING
    et_ltap_vb    = DATA(lt_created)
    et_bapiret    = DATA(lt_return)
    ev_severity   = DATA(lv_severity).

IF lv_severity CA 'EA'.
  " roll back, do not commit
ELSE.
  CALL FUNCTION '/SCWM/TO_POST'.       " [VERIFY] posting step
  COMMIT WORK AND WAIT.
ENDIF.
```

Three rules that matter more than the syntax:

1. **Never let the API commit for you** in code that participates in a larger LUW. Set `iv_commit_work = abap_false`, check severity, then commit explicitly. Getting this wrong produces partial postings that are extremely hard to diagnose.
2. **Always check `ev_severity` and the return table.** EWM APIs frequently return errors without raising exceptions.
3. **Understand whether you need the separate post step** in your release. Creating in memory and posting are distinct operations in several EWM APIs.

5.3 Confirming and cancelling

```
CALL FUNCTION '/SCWM/TO_CONFIRM'
  EXPORTING
    iv_lgnum       = lv_warehouse
    iv_commit_work = abap_false
    it_conf        = lt_confirm
  IMPORTING
    et_bapiret     = lt_return
    ev_severity    = lv_severity.
```

Confirmation moves the task from `ORDIM_O` to `ORDIM_C`. Partial confirmation, differences, and exception codes are all handled here — exception codes are the mechanism by which a worker reports “bin empty” or “damaged stock,” and they drive follow-on processes. Custom RF transactions must handle them or the warehouse loses its exception handling.

5.4 Warehouse Order Creation Rules

WOCR decides how tasks are bundled into orders. It is customizing, with BAdI extension points for logic that configuration cannot express.

Determination considers: activity area, warehouse process type, the limit values (number of items, weight, volume, time), sorting, and filters.

The architect's diagnostic sequence when order formation is wrong:

1. Is the packaging specification producing the expected task quantities? (Chapter 3.2 — check this *first*, it is the usual cause)
2. Is the activity area assignment correct for the bins involved?
3. Which WOCR was selected, and does its limit configuration match expectation?
4. Only then look for a BAdI implementation.

Working that sequence in order saves days. Most people start at step 4.

Chapter 6: Storage Control

The mechanism that turns a single logical movement into a multi-step physical process. This is EWM's most conceptually distinctive feature and the one most often misunderstood.

6.1 The two kinds

Process-oriented storage control (POSC). Breaks a movement into process steps: unload → count → deconsolidate → putaway. Each step is a separate warehouse task. Driven by the warehouse process type and the external process step definition.

Layout-oriented storage control (LOSC). Handles physical routing constraints — you cannot get from A to B directly, so the movement is broken at an identification point or a pick point. Driven by the warehouse layout, not the process.

They can both apply to the same movement. When a movement produces more tasks than expected, determine which control is responsible before changing anything.

6.2 Why architects must understand it

Because the wrong answer to a business requirement is usually custom code where storage control would have done it. "We need a quality check between goods receipt and putaway" is a POSC step, not a development.

The design rule: when a requirement is "add a step to a physical process," the first hypothesis is storage control. Reach for code only after proving configuration cannot express it. This single habit will differentiate you from most technical consultants entering EWM.

6.3 Where custom logic legitimately attaches

- Determining the process step sequence dynamically, where configuration cannot

- Deciding the intermediate destination at an identification point
- Skipping a step under specific conditions

All via BAdIs (Chapter 14), never by intercepting task creation.

Chapter 7: Handling Units

7.1 The model

An HU is a physical unit with an identifier, holding stock or other HUs. Header and items, with nesting. Almost every warehouse process either creates, moves, repacks or dissolves HUs.

7.2 Reading and working with HUs

```
" [VERIFY] names against your release
CALL FUNCTION '/SCWM/HU_SELECT_GEN'
  EXPORTING
    iv_lgnum    = lv_warehouse
    it_huident  = lt_huident
  IMPORTING
    et_huhdr    = DATA(lt_huhdr)
    et_huitm    = DATA(lt_huitm)
  EXCEPTIONS
    error       = 1
    OTHERS      = 2.
```

For packing operations, the HU packing API and the pack object are the correct route. Do not manipulate HU tables directly — HU consistency spans several tables and stock records, and direct writes corrupt it in ways that surface weeks later as unexplainable stock differences.

7.3 The architect’s HU questions

- **Identifier strategy.** Internal numbering, SSCC/GS1, or supplier HU identifiers adopted at goods receipt? This decision affects labels, scanning, and interfaces to customers and carriers.
- **Nesting depth.** How deep do HUs nest, and does every process handle that depth?
- **Supplier HUs.** Do you adopt them or repack? Adoption saves labour and couples you to supplier data quality.
- **Label design and printing.** Which content, which trigger, which printer determination. Handled through PPF (Chapter 15).

Chapter 8: Stock and Physical Inventory

8.1 The stock model

EWM stock is more granular than ERP stock. A quant is identified by combinations of product, batch, stock type, owner, party entitled to dispose, storage bin and handling

unit. Two stock records for the same material in the same bin are normal and correct if any other dimension differs.

`/SCWM/AQUA` is the available quantity table and the one you will read most.

8.2 Reading stock

```
" [VERIFY]
CALL FUNCTION '/SCWM/AQUA_SELECT'
  EXPORTING
    iv_lgnum  = lv_warehouse
    it_matid  = lt_matid
  IMPORTING
    et_aqua   = DATA(lt_stock).
```

The performance trap: stock selects without a bin, product or HU restriction scan very large tables. In a large warehouse `/SCWM/AQUA` holds millions of rows. Every custom stock read must be restricted, and every stock report must be paged.

8.3 Stock consistency between EWM and ERP

In a decentralized landscape, EWM stock and ERP stock are separate records reconciled by goods movements. They *will* diverge when a goods movement fails mid-flight.

As architect you own:

- The reconciliation report and its cadence
- The design for handling a failed goods movement — retry, manual correction, or compensating posting
- Monitoring that surfaces divergence before month end rather than during it

This is a design deliverable, not an operational afterthought, and it is routinely missing from EWM designs.

8.4 Physical inventory

EWM physical inventory supports periodic, continuous, low-stock-triggered and cycle counting. Technically, custom work usually attaches to:

- Document creation logic — which bins get counted, when
- Count entry via RF (Chapter 16)
- Difference analysis and approval workflow
- The posting of differences to ERP

Part III — Business Scenarios

Every scenario follows the same template: what the business wants, how standard EWM solves it, what a technical architect must know, where custom work usually lands, and the trap.

Use this as a lookup. In a workshop, find the scenario, read the standard solution before proposing anything, and check the pitfall before estimating. The most common EWM design failure is developing something that configuration already does.

All configuration object names are conceptually stable. Repository object names remain **[VERIFY]** per the warning in the front matter.

Chapter 9: Inbound Scenarios

S-01 Standard goods receipt and putaway

Business. Purchase order arrives, is received, and stock is put away.

Standard solution. ERP PO → inbound delivery → distributed to EWM → unload → goods receipt posting → putaway warehouse tasks via storage type search and bin determination.

Technical touchpoints. Delivery read API; warehouse process type for putaway; storage type search sequence; bin determination; packaging specification driving task quantities; WOCR bundling; goods movement to ERP.

Common enhancements. Custom bin determination rules; custom validation at GR; additional fields on the delivery propagated from ERP; custom GR label.

Pitfall. Custom bin determination BAdIs written without a process-type filter fire on every putaway in the warehouse. Filter early (Chapter 29.4).

S-02 Goods receipt with deconsolidation

Business. A mixed pallet arrives containing products destined for different storage sections; it must be split before putaway.

Standard solution. Process-oriented storage control with a deconsolidation step. EWM routes the HU to a deconsolidation work centre, the operator splits it, and putaway tasks are then created per destination.

Technical touchpoints. POSC configuration; work centre definition; deconsolidation determination (which HUs need it); RF or work centre UI; HU repacking API.

Common enhancements. Custom rules for whether an HU needs deconsolidation; custom work centre screens; label printing per resulting HU.

Pitfall. Teams build a custom “split pallet” development without discovering that deconsolidation is standard POSC. Always check storage control first (Chapter 6.2).

S-03 Quality inspection at goods receipt

Business. Received stock must be inspected before it becomes available.

Standard solution. Quality inspection integration. Stock is received into a non-available stock type, an inspection document is created, and the usage decision triggers a posting change to available or to blocked/returns.

Technical touchpoints. Stock types and their meaning; inspection object types; posting change process; the QM integration interface; warehouse process types for the follow-on movement.

Common enhancements. Custom sampling rules; custom inspection triggers by vendor or product; RF transaction for inspection recording (Chapter 30).

Pitfall. Stock type handling. Custom code that reads available stock without filtering stock type will silently include or exclude inspection stock. This is a frequent source of wrong stock reports.

S-04 Cross-docking

Business. Received goods go directly to shipping rather than into storage.

Standard solution. Several variants — opportunistic cross-docking decided at goods receipt, planned cross-docking decided in advance, and transportation cross-docking. Each has its own determination and routing.

Technical touchpoints. Cross-docking determination logic; the relationship between inbound and outbound deliveries; storage control routing to the staging area; warehouse process types for cross-dock movements.

Common enhancements. Custom cross-docking decision rules based on demand urgency; custom monitoring of cross-dock candidates.

Pitfall. Cross-docking couples inbound and outbound timing. A design that works when both deliveries exist fails when the outbound is created later. Design the “no matching demand” path explicitly.

S-05 Returns from customer

Business. Customer returns goods; they must be received, inspected, and dispositioned.

Standard solution. Returns delivery from ERP → EWM inbound processing with returns-specific stock types → inspection → posting change to available, blocked, or scrap.

Technical touchpoints. Returns document category; stock type transitions; inspection integration; the disposition decision and its posting.

Common enhancements. Custom disposition rules; RF transaction for returns grading; integration to a refurbishment process.

Pitfall. Returns processes accumulate exception handling. Scope the dispositions explicitly at design time; open-ended “and other cases” becomes unbounded development.

S-06 Unplanned or reference-free goods receipt

Business. Stock arrives with no purchase order or delivery.

Standard solution. Ad-hoc goods receipt in EWM, with the ERP posting created subsequently.

Technical touchpoints. Ad-hoc movement functions; the ERP goods movement interface; the reconciliation implications (Chapter 8.3).

Pitfall. This is the scenario most likely to create EWM/ERP stock divergence. Any design permitting it must specify the reconciliation and approval process.

S-07 Value-added services at inbound

Business. Received goods need labelling, kitting, assembly or repackaging before putaway.

Standard solution. VAS order, generated from a VAS-relevant condition, executed at a work centre with its own UI.

Technical touchpoints. VAS determination; VAS order structure; work centre configuration; auxiliary product consumption; the packaging specification's role.

Common enhancements. Custom VAS determination; custom work centre screens; consumption posting of auxiliary materials.

Pitfall. VAS effort is frequently under-modelled — auxiliary product consumption and labour capture are separate design decisions and both get forgotten until UAT.

S-08 Expected goods receipt from ASN

Business. Advance shipping notification lets the warehouse plan before physical arrival.

Standard solution. Expected goods receipt objects created from the ASN, converted to inbound deliveries on arrival.

Technical touchpoints. The EGR object; conversion logic; the relationship to dock appointment scheduling.

Pitfall. ASN data quality. Design the mismatch path — what happens when the physical delivery does not match the ASN — before designing the happy path.

Chapter 10: Outbound Scenarios

S-09 Standard picking to goods issue

Business. Sales order ships from the warehouse.

Standard solution. ERP outbound delivery → EWM outbound delivery order → wave → pick warehouse tasks → packing → staging → loading → goods issue.

Technical touchpoints. Outbound delivery order document category; wave determination; stock removal strategy; WOCR; storage control for the pick-pack-stage sequence; goods issue posting back to ERP.

Common enhancements. Custom stock removal rules; custom picking sequence; additional delivery fields; custom packing logic.

Pitfall. The outbound delivery order is EWM's planning document and does not map one-to-one to the ERP delivery. Code assuming a 1:1 relationship breaks on splits and merges.

S-10 Wave management

Business. Group orders into work batches aligned to shipping windows, carriers or zones.

Standard solution. Wave templates with determination criteria; automatic or manual wave assignment; wave release creating warehouse tasks.

Technical touchpoints. Wave template determination; wave release processing (mass, background, and a performance hotspot); the relationship between waves, activity areas and WOCR.

Common enhancements. Custom wave determination; custom release scheduling; wave monitoring dashboards.

Pitfall. **Wave release is the single biggest performance risk in outbound.** It processes large volumes in one run. Design for packaging and restartability (Chapter 20.1) and test at real volume.

S-11 Two-step picking

Business. Pick bulk quantities from storage once, then distribute to individual orders in a second step.

Standard solution. Two-step picking configuration — a removal step to a consolidation area, then an allocation step per order.

Technical touchpoints. Two-step relevance determination; the intermediate stock location; the link between the two steps; consolidation group logic.

Pitfall. Reconciling the intermediate stock when the second step is interrupted. Design the exception path.

S-12 Batch and FEFO picking

Business. Stock must be picked by expiry date, batch characteristics or country of origin.

Standard solution. Batch management with stock removal strategy driven by shelf life; batch determination on the delivery.

Technical touchpoints. Batch master data and its replication (decentralized); shelf life fields; stock removal strategy; the interaction between batch determination in ERP and stock removal in EWM.

Common enhancements. Custom batch selection rules; customer-specific minimum remaining shelf life.

Pitfall. Batch determination can happen in ERP or in EWM depending on configuration. Establish which, before designing anything that depends on it — this is a genuinely confusing area and getting it wrong invalidates the design.

S-13 Serial number management

Business. Individual units must be tracked by serial number.

Standard solution. Serial number profiles determining where serial numbers are captured — at goods receipt, at picking, at packing, or at goods issue.

Technical touchpoints. Serial number requirement levels; capture points; RF screens for serial capture; the volume implication of serial-level records.

Common enhancements. Custom serial capture screens; validation against supplier data; serial-level traceability reporting.

Pitfall. Serial numbers multiply data volume enormously. A design that captures serials at every step in a high-volume warehouse has a performance and storage consequence that must be modelled before commitment.

S-14 Kitting and kit-to-order

Business. Components are assembled into a kit for shipment.

Standard solution. Either VAS-based kitting or production integration, depending on whether the kit is a stocked material.

Technical touchpoints. Kit structure source; component staging; consumption posting; the resulting HU structure.

Pitfall. Choosing VAS versus production integration is an architecture decision with long consequences for costing and traceability. Decide it with the finance and production streams, not alone.

S-15 Packing and packing specifications

Business. Picked goods are packed into shipping HUs according to rules.

Standard solution. Packing at a work centre or during picking, driven by packaging specifications; automatic or manual HU creation.

Technical touchpoints. Packaging specification determination; HU hierarchy; packing work centre UI; shipping HU identifier strategy (SSCC).

Common enhancements. Custom packing rules; carton selection logic; weight and dimension capture integration.

Pitfall. SSCC number range exhaustion and identifier strategy. Decide the identifier approach at design time; changing it later affects labels, interfaces and customer EDI.

S-16 Staging and loading

Business. Packed goods move to a door and are loaded onto a vehicle.

Standard solution. Storage control routing to the staging area; loading via RF or the work centre; goods issue on load completion.

Technical touchpoints. Door and staging area determination; the transportation unit and vehicle objects; loading confirmation; the goods issue trigger.

Pitfall. Door determination logic frequently becomes custom. Confirm the standard determination cannot express the requirement before building.

S-17 Shipping and transportation integration

Business. Shipments must be planned, consolidated and tendered to carriers.

Standard solution. Integration with transportation management — freight orders drive warehouse execution timing; transportation units and vehicles are shared objects.

Technical touchpoints. The TU and vehicle model; the freight order interface; how transportation planning constrains wave and staging timing.

Pitfall. The sequencing dependency between transportation planning and wave release. Designing warehouse execution without understanding it produces waves released for shipments that have no vehicle.

S-18 Direct outbound delivery and stock transfers

Business. Stock ships without a sales order, or moves between plants.

Standard solution. Direct outbound delivery order created in EWM, or an ERP stock transport order flowing through the standard outbound process.

Technical touchpoints. Document categories; the ERP posting; the receiving side if it is also EWM-managed.

Pitfall. Intra-company transfers where both ends are EWM warehouses need the goods issue and goods receipt designed as one flow, not two projects.

Chapter 11: Internal and Stock Scenarios

S-19 Replenishment

Business. Picking bins must be refilled from reserve storage before they run empty.

Standard solution. Several types — planned replenishment run on min/max, order-related replenishment triggered by demand, automatic replenishment triggered at pick confirmation, and direct replenishment.

Technical touchpoints. Replenishment control data on the storage bin and warehouse product; the replenishment run and its scheduling; the warehouse process type for the movement; how automatic replenishment hooks into pick confirmation.

Common enhancements. Custom min/max calculation from demand history; replenishment priority rules; monitoring of replenishment shortfalls.

Pitfall. Choosing the wrong replenishment type. Automatic replenishment at pick confirmation adds work to the pick transaction and can slow the picking process measurably. Model the timing consequence, not just the logic.

S-20 Ad-hoc and internal movements

Business. Move stock for operational reasons — consolidation, housekeeping, clearing a blocked bin.

Standard solution. Ad-hoc warehouse task creation via RF or the monitor, using a dedicated warehouse process type.

Technical touchpoints. Ad-hoc task creation API (Chapter 5.2); process type configuration determining what is permitted; authorisation.

Pitfall. Ad-hoc movement is a governance question as much as a technical one. Who may move what, and is it logged? Design the authorisation model deliberately or the warehouse becomes untraceable.

S-21 Posting changes

Business. Stock changes status, owner, batch or stock type without moving.

Standard solution. Posting change process, with or without a physical movement, communicated to ERP as a stock transfer posting.

Technical touchpoints. Posting change document; stock type transitions; the ERP interface; whether a physical movement accompanies the change.

Pitfall. Posting changes that fail on the ERP side leave EWM and ERP divergent. This is a primary contributor to reconciliation problems (Chapter 8.3).

S-22 Scrapping

Business. Damaged or expired stock is written off.

Standard solution. Scrapping process using a dedicated process type and stock type, with an ERP goods movement.

Technical touchpoints. Approval workflow if required; the ERP posting; reason codes.

Pitfall. Approval requirements are usually discovered late. Ask about authorisation thresholds during design, not during UAT.

S-23 Physical inventory

Business. Count stock to verify system accuracy.

Standard solution. Periodic, continuous, cycle counting, low-stock-triggered and zero-stock-check variants. Documents created, counted, differences analysed and posted.

Technical touchpoints. PI document creation logic; count entry via RF; difference tolerance and approval; the ERP posting of differences; the relationship to the fiscal year requirement.

Common enhancements. Custom document creation rules; custom RF count screens; difference approval workflow; recount logic.

Pitfall. The difference posting to ERP is where PI designs fail. Confirm the posting mechanism, the tolerance handling and the accounting treatment with the finance stream.

S-24 Slotting and rearrangement

Business. Optimise which products sit in which bins based on demand and physical characteristics.

Standard solution. Slotting determines the optimal storage parameters; rearrangement generates the movements to realise them.

Technical touchpoints. Slotting data sources — demand, product dimensions, packaging specifications; the slotting run; rearrangement task generation.

Pitfall. Slotting is frequently sold and rarely used, because it needs reliable dimension and demand data most warehouses do not have. Verify data quality before committing to it in a design.

S-25 Stock consolidation and housekeeping

Business. Combine partial quantities to free bins and improve density.

Standard solution. Rearrangement, or ad-hoc movements driven by a report.

Technical touchpoints. Bin capacity data; consolidation candidate identification; task creation in bulk.

Pitfall. Housekeeping movements compete with productive work for resources. Design the queue and priority handling or picking slows down at peak.

Chapter 12: Manufacturing Integration

S-26 Staging to production

Business. Components must be delivered to a production supply area before a production order runs.

Standard solution. Several staging types — pick parts staged per order, crate parts, release order parts, and single-order staging. Driven by the production supply area and the staging indicator on the material.

Technical touchpoints. Production supply area model; the ERP production order interface; staging warehouse process types; the relationship between the production order and the warehouse request.

Common enhancements. Custom staging quantity rules; staging call-off timing; integration to a line-side replenishment signal.

Pitfall. Timing. Staging too early consumes line-side space; too late stops production. The trigger design is a business decision that must be settled before technical design.

S-27 Receipt from production

Business. Finished goods from production are received into the warehouse.

Standard solution. Goods receipt from a production order, with HU creation and putaway.

Technical touchpoints. The production order interface; automatic HU creation via packaging specification; putaway determination; backflush implications.

Pitfall. HU creation at production receipt frequently needs custom identifier logic to match production's own labelling. Establish the identifier strategy early (Chapter 7.3).

S-28 Advanced production integration

Business. Tighter coupling between EWM and production execution, with component consumption and order confirmation flowing through the warehouse.

Standard solution. Advanced production integration, which extends the basic staging model with a closer link to production execution.

Technical touchpoints. The extended interface; consumption posting; the production material request object.

Pitfall. This is a significant scope decision, not a configuration switch. Assess whether the client's production process genuinely needs it — basic staging covers many cases at a fraction of the complexity.

S-29 Kanban and line-side replenishment

Business. Production pulls material as it is consumed.

Standard solution. Kanban integration triggering replenishment movements to the supply area.

Technical touchpoints. Kanban signal handling; replenishment task creation; the physical scan or button that triggers it.

Pitfall. Signal latency. Design the timeout and escalation path for a Kanban signal that produces no replenishment.

Chapter 13: Cross-Functional and Third-Party Logistics

S-30 Resource and queue management

Business. Work must be distributed to operators and equipment efficiently.

Standard solution. Resources, resource types, resource groups, queues and queue determination. Warehouse orders are assigned to queues; resources pull from queues by priority.

Technical touchpoints. Queue determination logic; resource assignment to queues; the relationship between activity area, queue and WOCR; how the RF transaction selects the next warehouse order.

Common enhancements. Custom queue determination; custom order selection logic; skill-based assignment.

Pitfall. Queue design is the difference between a productive and an unproductive warehouse, and it is usually treated as an afterthought. It deserves the same design attention as storage control.

S-31 Labor management

Business. Measure and manage operator productivity against engineered standards.

Standard solution. Labor management with engineered labor standards, executed workload capture, and performance reporting.

Technical touchpoints. Activity capture from warehouse task confirmation; standard definition; the indirect labor capture mechanism; the reporting layer.

Common enhancements. Custom standards; integration to payroll or incentive schemes.

Pitfall. Labor management is an industrial-relations question before it is a technical one. Do not design incentive-linked measurement without confirming the client has the agreements in place.

S-32 Yard and dock appointment scheduling

Business. Manage vehicles from arrival at the gate to departure.

Standard solution. Yard management with yard movements as warehouse tasks; dock appointment scheduling for slot booking.

Technical touchpoints. Transportation unit and vehicle objects; yard bins as storage bins; check-in and check-out processing; the appointment interface, often to carriers.

Pitfall. Carrier-facing appointment booking usually needs an external portal. That is an integration project, not an EWM configuration.

S-33 Third-party logistics and warehouse billing

Business. A 3PL operator must bill clients for storage and handling.

Standard solution. Warehouse billing measurement services capturing billable activities, aggregated and passed to billing.

Technical touchpoints. Measurement service definition; the activity capture points; the interface to the billing system; party entitled to dispose as the client separator.

Common enhancements. Custom measurement services; client-specific rate handling; billing reconciliation reports.

Pitfall. In 3PL, **party entitled to dispose is the central data separator** and it must be correct in every custom object, report and interface. A custom report that ignores it leaks one client’s data to another — a contractual incident, not a bug.

S-34 Automated warehouse execution

Business. Conveyors, AS/RS and sorters execute movements without operators.

Standard solution. MFS (Chapter 17), with communication points, telegrams and conveyor segment modelling.

Technical touchpoints. All of Chapter 17.

Pitfall. The boundary with the equipment vendor's own control system. Establish in writing who owns sequencing (Chapter 19).

S-35 Multi-warehouse and multi-ERP landscapes

Business. One EWM system serves several warehouses, or several ERP systems.

Standard solution. Multiple warehouse numbers in one EWM client; business system group configuration for multiple ERP connections.

Technical touchpoints. Warehouse number as the primary partitioning key; business system group; document number ranges; master data replication from multiple sources.

Pitfall. **Every custom object must be warehouse-number aware.** Code written against a single warehouse during a pilot and then rolled out breaks in ways that are hard to find. Make `lgnum` a mandatory parameter everywhere from day one, even when there is only one warehouse.

S-36 Interfacing an external WMS or WCS

Business. Part of the warehouse is controlled by a third-party system.

Standard solution. Interface at an agreed boundary, usually task-level or order-level.

Technical touchpoints. The boundary definition; message design; stock reconciliation between the two systems; error handling.

Pitfall. Duplicated logic on both sides of the boundary. Whichever system owns sequencing must own it completely.

13.1 The scenario-to-technical-area map

Use this to see which technical skills a given project actually needs.

Scenario group	Config depth	RF	PPF	BAdI	MFS	Integration
Inbound (S-01 to S-08)	High	High	High	High	Low	Medium
Outbound (S-09 to S-18)	High	High	High	High	Low	High
Internal (S-19 to S-25)	High	Medium	Low	Medium	Low	Medium
Manufacturing (S-26 to S-29)	Medium	Medium	Low	Medium	Low	High
Cross-functional (S-30 to S-36)	High	Medium	Medium	High	Variable	High

The reading: configuration depth is high everywhere. That is the core message of this part. An EWM technical architect who cannot configure is guessing at every design decision, which is why Chapter 31.1 puts configuration literacy before the extension framework in the learning sequence.

Part IV — The Extension Framework

Chapter 14: Where to Hook In

14.1 The extension hierarchy

Work down this list. Stop at the first that works. This ordering is the single most valuable discipline in EWM development, because EWM's configuration layer is unusually powerful and developers habitually skip past it.

1. **Configuration.** Warehouse process type, storage control, WOCR, determination procedures, condition technique. A large share of "development requirements" are unconfigured configuration.
2. **Packaging specification / condition records.** Data-driven behaviour.
3. **PPF action.** Output, printing, triggering follow-on processing.
4. **BAdI in the `/SCWM/` namespace.** The designed extension point.
5. **Custom object consuming EWM APIs.** Your own program calling standard function modules.
6. **Enhancement / modification.** Never.

14.2 The BAdI landscape

EWM ships a large, deliberately structured BAdI set. The families you will use most:

Area	What you can influence
Delivery	Field mapping, validation, status determination, distribution
Warehouse task creation	Source and destination determination, quantity splitting, process type
Warehouse order creation	Bundling, sorting, limit evaluation
Storage control	Step sequence, intermediate destinations
Handling unit	Identifier assignment, packing behaviour
Physical inventory	Document creation, difference handling
RF	Screen flow, field validation, custom logic per step
Goods movement	Posting data, ERP interface content

[VERIFY] every specific BAdI name. Find them by browsing the enhancement spots under the EWM IMG node for the relevant process, not from memory.

14.3 How to find the right BAdI

The method, which works on any release and is more valuable than any list:

1. Identify the exact moment in the process where you need to intervene.
2. Run the process with a breakpoint on `CL_EXITHANDLER=>GET_INSTANCE` — or the modern equivalent for your release — and observe which BAdIs are called at that moment.
3. Read the BAdI's interface: what data does it receive, what can it change?
4. Confirm it is called on **every** path that matters — RF, background, API, Fiori. A BAdI called only from the dialog path is a defect waiting for the first background run.

Step 4 is the one people skip and it is the source of the classic EWM defect: "the validation works when the operator scans it but not when the wave releases it."

14.4 Clean core in EWM

EWM is one of the most heavily customized areas in any S/4HANA landscape, and released API coverage in EWM is thinner than in finance or sales. Practical position:

- Check the API State of every `/SCWM/` object you consume. Some are released; many are not.
- Where a BAdI is not released, apply the tier-2 pattern: implement the hook, delegate immediately to a cloud-compliant class holding the logic.
- Register every wrapper with a replacement condition.

This is your differentiating position. Clean-core extensibility *inside* EWM is a question almost nobody can answer well, and every WM-to-EWM migration raises it.

Chapter 15: The Post Processing Framework

PPF is EWM's output and follow-on action mechanism. Labels, documents, and triggered processing all run through it.

15.1 The model

- **Action profile** — attached to an application object (delivery, warehouse order, HU)
- **Action definition** — a specific thing that can happen
- **Schedule condition** — should this action be planned?
- **Start condition** — should it execute now?
- **Action method / processing type** — what it actually does: smart form, Adobe form, method call

15.2 A method-based action

```
CLASS zcl_ewm_ppf_label DEFINITION PUBLIC FINAL.
  PUBLIC SECTION.
    INTERFACES if_ex_exec_methodcall_ppf.   " [VERIFY] interface name
ENDCLASS.

CLASS zcl_ewm_ppf_label IMPLEMENTATION.
  METHOD if_ex_exec_methodcall_ppf~execute.
    " io_appl_object gives access to the application object
    " Keep this thin: extract context, delegate, return status
    TRY.
        DATA(lo_handler) = NEW zcl_ewm_label_handler( ).
        lo_handler->print( io_appl_object ).
        ep_status = 1.                       " [VERIFY] status constants
      CATCH zcx_ewm_label INTO DATA(lx).
        ep_status = 2.
    ENDTRY.
  ENDMETHOD.
ENDCLASS.
```

15.3 Design rules for PPF

- **Schedule conditions in configuration, not in the method.** A method that starts with “if this is not the right case, return” is a schedule condition written in the wrong place, and it is invisible to anyone reading the configuration.
- **Keep the method thin.** Extract context, delegate to a testable class. PPF methods are hard to unit test; your logic class should not be.
- **Never let a PPF failure fail the business process** unless it genuinely should. A label printer being offline should not block a goods issue — decide this deliberately per action, and document the decision.
- **Printer determination is a design decision**, not a hardcoded value. Bin, activity area, resource and workstation are all legitimate determination bases.

15.4 The common failure

Labels not printing is the most frequent EWM production incident. The diagnostic order: was the action scheduled → was the start condition met → did the method execute → did the spool request generate → did the device receive it. Debugging from the printer backwards wastes hours; debugging from the schedule condition forwards takes minutes.

Part V — RF and MFS

These two chapters are what make an EWM developer scarce. Most ABAP developers entering EWM never learn either properly.

Chapter 16: The RF Framework

16.1 Why it exists

Warehouse operators work on handheld scanners and vehicle-mounted terminals: small screens, keyboard-driven, barcode input, no mouse, high transaction volume, poor network conditions. The RF framework provides a screen and flow model designed for exactly that, rendered through a lightweight UI layer.

It is not Dynpro, not Fiori, and not Web Dynpro. Your existing UI skills do not transfer, which is precisely why this skill is scarce and defensible.

16.2 The architecture

The layers, from configuration down:

- **Logical transaction** — a business activity, e.g. picking by warehouse order
- **Step** — one interaction within the transaction
- **Screen** — the physical layout, in variants for different device sizes
- **Function code** — what a key or a scan triggers
- **Presentation profile / display profile** — device-specific rendering
- **Personalisation** — per user or resource

Navigation between steps is **configuration-driven**, not coded. This is the mental shift: you configure the flow and code the logic behind each step.

16.3 Building a custom RF transaction

The sequence — the detail differs by release, so **[VERIFY]** the customizing paths, but the order is stable:

1. **Define the logical transaction** and register it in RF customizing.
2. **Define the steps** and the flow between them, including error paths.
3. **Create the screens.** Copy an SAP screen close to your need rather than starting blank — you inherit correct sizing and conventions.
4. **Create the function module or class per step** that holds the logic.
5. **Assign function codes** and map keys.
6. **Register the transaction in the RF menu** for the relevant roles.
7. **Test on the actual device profile**, not only in the desktop simulator.

16.4 A step handler

```
FUNCTION z_rf_pick_confirm_qty.
* [VERIFY] the exact framework contract for your release —
* parameter names, the global data area, and the return-code convention
* all differ between EWM 9.x and embedded S/4HANA EWM.

  " 1. Read the current screen fields from the RF work area
  " 2. Validate the scan
  IF lv_scanned_hu IS INITIAL.
    " Framework message call — do not use MESSAGE directly,
    " the RF layer must control display and navigation
    PERFORM message_display USING 'ZEWM' 'E' '001'.
    " set the field to re-focus, stay on this step
    RETURN.
  ENDIF.

  " 3. Delegate the actual business logic to a testable class
  TRY.
      DATA(lo_pick) = NEW zcl_ewm_pick_service( ).
      lo_pick->confirm(
        iv_lgnum = gv_lgnum
        iv_tanum = gv_tanum
        iv_quan  = gv_quan_scanned ).
    CATCH zcx_ewm_pick INTO DATA(lx).
      PERFORM message_display USING lx->get_message( ).
      RETURN.
  ENDTRY.

  " 4. Set the next step / function code for the framework
  gv_fcode = 'NEXT'.
ENDFUNCTION.
```

16.5 The rules that separate good RF work from bad

1. **No business logic in the step handler.** Delegate to a class. RF handlers are close to untestable; classes are not. This one rule accounts for most of the quality difference between EWM developers.
2. **Never use bare `MESSAGE`.** The framework owns display and navigation. A raw message breaks the flow and can leave the device stuck.
3. **Design the error path as carefully as the happy path.** An operator with a scanner and a wrong barcode needs a clear, recoverable state. Most RF defects are error-path defects.
4. **Minimise round trips.** Every step is a network round trip on warehouse WiFi. A five-step transaction where three steps could be one is a throughput problem multiplied by thousands of picks per shift.
5. **Keep the scan the primary input.** If the operator has to type, the design is wrong.
6. **Test on the real device profile.** Screen sizes differ; a field that fits the simulator may not fit the terminal.
7. **Handle exception codes.** Bin empty, damaged stock, short pick. These drive follow-on processes and omitting them silently removes the warehouse's exception handling.

16.6 The architect’s RF questions

- How many steps per transaction, and what is the target seconds-per-pick?
- What happens when the network drops mid-transaction?
- Which exceptions can the operator report, and what does each trigger?
- Is this transaction a copy of a standard one with small changes? If so, can configuration and a BAdI achieve it instead of a full custom transaction?

That last question is the most valuable. **Many custom RF transactions should have been a standard transaction plus a BAdI.** A copied transaction is a permanent maintenance liability that no longer receives SAP’s improvements.

Chapter 17: Material Flow System

17.1 What MFS is

MFS makes EWM the warehouse control system for automated equipment: conveyors, automated storage and retrieval machines, sorters. EWM talks directly to PLCs rather than through a third-party WCS.

This is the highest-scarcity area in EWM development. It is also the highest risk: a defect stops physical machinery.

17.2 The model

- **Communication point / channel** — the connection to a PLC
- **Telegram** — the message exchanged, with a defined structure
- **Conveyor segment and resource** — the modelled physical layout
- **Communication point actions** — what happens when stock reaches a point

Flow: EWM creates a warehouse task → a telegram instructs the PLC → the PLC moves the load → the PLC sends a status telegram → EWM confirms or reacts.

17.3 Custom telegram handling

Custom telegram types and handling logic are the common development. Design rules:

- **Telegram structures are contracts with the PLC vendor.** Version them. Changing a structure without coordinating is a physical outage.
- **Every telegram path needs a timeout and an error state.** A load stuck at a communication point with no timeout blocks the conveyor.
- **Log everything.** MFS troubleshooting is forensic; without a telegram log you cannot reconstruct what happened.
- **Design for PLC restart.** The PLC will reboot. What is the resynchronisation procedure, and who runs it?
- **Never block the telegram handler.** Long-running logic in a telegram path backs up the queue and stops the equipment.

17.4 Testing MFS

You cannot test against live equipment. You need:

- A telegram simulator that plays recorded and synthetic sequences
- Test cases for every error path: timeout, wrong load, blocked segment, PLC restart, duplicate telegram
- An agreed integration test window with the equipment vendor
- A rollback plan that leaves the warehouse operable manually

If a client cannot provide a simulator, that is a project risk to raise in writing before you start. Discovering it at integration test costs weeks.

17.5 Whether to specialise here

MFS is the scarcest EWM skill and commands the highest rates. It also ties you to automated warehouse projects, which are fewer and concentrated in specific industries. It is a genuine specialisation decision, not an add-on — treat it as a second bet to make after you are established in mainstream EWM technical work, not a first one.

Part VI — Integration

Chapter 18: ERP to EWM

18.1 Embedded versus decentralized, technically

Concern	Embedded	Decentralized
Delivery transfer	Internal, synchronous	Queued, asynchronous
Master data	Shared tables	Replicated
Goods movement	Direct posting	Posted in EWM, communicated to ERP
Failure modes	Few	Queue blocks, mapping errors, replication lag
Monitoring surface	Small	Large — and often under-designed

The architect's summary: embedded removes an entire class of problems. Decentralized buys independence at the cost of an integration layer you must design, monitor and operate. Choose on the availability requirement, not on preference.

18.2 What you own in a decentralized design

Deliverables that are routinely missing and that you should insist on:

- **Queue monitoring and alerting.** Which queues, what threshold, who is alerted, within what time.
- **Error handling per message type.** Retry, manual repair, or reject — decided per type, not generically.
- **Reprocessing procedure.** After a fix, how does the stuck message get reprocessed, and by whom?
- **Master data replication monitoring**, including custom field extensions.
- **Stock reconciliation** between EWM and ERP, with cadence and ownership (Chapter 8.3).
- **Startup and shutdown sequencing** for planned ERP downtime.

18.3 Custom field propagation

A custom field on the ERP delivery that must reach EWM crosses several boundaries: ERP structure → distribution message → EWM delivery structure → EWM database → possibly the RF screen and the label.

Each boundary is a separate extension. **Estimate this as five pieces of work, not one.** Under-estimating custom field propagation is the most common EWM estimation error, and it is visible in almost every project retrospective.

Chapter 19: External Systems

System	Interface	Design notes
PLC / automation	MFS telegrams	Chapter 17
Third-party WCS	Usually IDoc or web service	Decide the boundary explicitly: who owns task sequencing?
Label printers	PPF → spool → device	Determination logic is a design decision
Transport / carrier	API or file	Often via an integration layer
Yard / dock scheduling	API	Frequently a separate product
Weighing, dimensioning	Device integration	Often through the RF layer

The boundary question for any automation integration: does EWM sequence the work and the external system execute, or does the external system own sequencing? Ambiguity here causes duplicated logic, conflicting instructions, and blame in production. Settle it in the architecture document, in writing, signed by both vendors.

Part VII — Quality

Chapter 20: Performance in EWM

20.1 Where EWM performance problems actually live

Area	Typical cause
Stock selection	Unrestricted `/SCWM/AQUA` reads on millions of rows
Warehouse task creation	Determination logic in a BAdI doing per-item database reads
Wave release	Mass processing without packaging or parallelisation
RF round trips	Too many steps, each a network round trip
Delivery query	`head_deep` / `item_deep` requesting everything
WOCR	Complex sorting over large task sets
Monitor reports	Unbounded selection with no paging

20.2 The rules

- Every stock read restricted by bin, product, HU or a combination
- No database read inside a loop over warehouse tasks or delivery items — read the set once
- Wave and mass processing in packages, restartable
- RF steps minimised; measure seconds per pick, not milliseconds per statement
- Delivery reads request only the data actually used
- Test at real warehouse volume: a design that works on 200 tasks may fail on 20,000 in a wave

20.3 The metric that matters

Not statement runtime. **Picks per operator hour**, and **wave release duration against the shipping window.** Those are the numbers the business feels. Tune toward them, and state the target before you start.

Chapter 21: Testing and Debugging

21.1 The first diagnostic question

Is this a code problem or a customizing problem? In EWM the answer is customizing far more often than in ordinary ABAP. Before debugging:

1. Which warehouse process type was used, and what does it configure?
2. Which storage control applied?
3. Which WOCR was selected?
4. What did packaging specification determination produce?
5. Only then: which BAdI implementations are active?

21.2 Techniques

- **SQL trace over a process step** — the fastest way to see what EWM actually touched, and the best way to learn the data model
- **Breakpoint on the BAdI handler** to enumerate which BAdIs fire when
- **The EWM warehouse management monitor** — before debugging, look at the object state in the monitor; it often answers the question outright
- **Application log** for process errors
- **Telegram log** for MFS
- **RF simulator** for screen flow, then the real device for layout

21.3 Testing strategy

Level	What
Unit	Logic classes behind RF handlers, BAdIs and PPF methods, with test doubles
Component	Warehouse task creation and confirmation against a test warehouse
Process	Full inbound and outbound cycles, including exceptions
Volume	Wave release and mass confirmation at production volume
Device	Every RF transaction on every device profile in use
Integration	ERP↔EWM including queue failure and recovery
Equipment	MFS against a simulator, then a vendor integration window

Build a test warehouse with representative structure, stock and master data, and script its rebuild. EWM testing without a reproducible warehouse state produces results nobody can trust.

Part VIII — The Architect’s Deliverables

Parts I to VII teach you to build in EWM. This part is what an architect is accountable for that a developer is not. Each chapter is a deliverable you produce, not a technique you apply.

Chapter 22: Warehouse Structure Design

The first deliverable on any EWM project, and the only one that is genuinely irreversible once stock exists. Get it wrong and every later design decision inherits the mistake.

22.1 The modelling hierarchy

Physical building, mapped downward:

Level	Object	What it represents
Site	Warehouse number	The warehouse as a whole — the top partitioning key
Zone	Storage type	An area with common storage behaviour: racking, bulk, cold, staging
Sub-zone	Storage section	A subdivision within a storage type, usually for putaway strategy
Location	Storage bin	The addressable place stock sits
Work zone	Activity area	A grouping of bins for the purpose of organising work
Position	Work centre	Where an operator performs an activity: packing, deconsolidation, VAS
Boundary	Door, staging area	Interface to the outside world

22.2 The decisions and their consequences

One warehouse number or several?

One warehouse number per physical site is the default. Use several when the site contains genuinely independent operations with different processes, different parties entitled to dispose, or different operating hours.

Consequence: stock cannot move between warehouse numbers as a simple warehouse task — it becomes a posting change or a stock transfer. Splitting a site across warehouse numbers to “keep things tidy” creates permanent friction in every internal movement.

Storage type granularity.

A storage type should represent a distinct *behaviour*, not a distinct physical area. Two racking areas with identical putaway and removal behaviour are one storage type with two sections, not two storage types.

Consequence: over-granular storage types multiply the storage type search sequence and make putaway determination hard to reason about. Under-granular types force behaviour into custom code.

Activity area design.

Activity areas group bins for work organisation, and they drive WOCR and queue assignment. They are independent of storage types and this trips people up constantly.

Consequence: this is the design that determines whether pickers walk efficiently. It is an industrial-engineering decision expressed in configuration, and it is where a warehouse becomes productive or does not.

22.3 Bin naming — the decision people regret

Bin identifiers encode the physical coordinate system, and sorting for pick path optimisation depends on them.

Rules that hold up:

- **Encode the physical coordinates** — aisle, stack, level — in a fixed-width scheme with leading zeros. `A01-05-3` sorts correctly; `A1-5-3` does not.
- **Fixed width everywhere.** Variable-length segments break sorting the day the warehouse extends past bin 9.
- **Leave room for extension.** Two digits for aisles when you have eight today and a planned expansion is not paranoia; renaming bins after go-live means relabelling a building.
- **Do not encode meaning that changes.** A bin identifier containing the product family is wrong the first time you re-slot.
- **Match the physical labels.** The bin identifier in the system must be what is printed on the rack. Obvious, and it gets missed on extensions.

Consequence: pick path sequencing is driven by sorting on this identifier. A scheme that does not sort in walking order means every pick route is wrong and no amount of WOCR configuration fixes it.

22.4 Doors, staging areas and yard

- Doors are the physical interface. Determination logic assigns deliveries to doors.
- Staging areas are storage locations with a relationship to doors.
- Yard bins are storage bins in a yard storage type; vehicles move between them as warehouse tasks.

Design question: is a door dedicated to inbound, outbound, or both, and does that change by shift? Dual-use doors need determination logic that considers time.

22.5 The deliverable

A warehouse structure document containing:

- The site layout with the logical mapping overlaid
- Warehouse number, storage type and section list with the behaviour rationale for each
- Activity area design with the work-organisation rationale
- Bin naming convention with worked examples and the extension allowance
- Door, staging area and work centre placement
- The bin master data load specification

Sign-off from operations, not just IT. The people who walk the aisles will find the problems that a floor plan does not show.

Chapter 23: Sizing, Volumetrics and Non-Functional Requirements

The numbers you must gather before designing anything, and the ones you will be held to at go-live.

23.1 The volumetrics to collect

Get these from the business as *numbers*, not adjectives. “High volume” is not a requirement.

Metric	Why it matters
Deliveries per day, inbound and outbound	Sizing the delivery and queue layers
Lines per delivery, average and peak	Warehouse task volume
Warehouse tasks per day	The primary throughput number
Picks per operator hour, target	The productivity number the design is judged on
Concurrent RF users, peak	Sizing and RF response requirements
HUs created per day	HU table growth, label volume
Wave size and wave count per day	Wave release performance (Chapter 20.1)
Bins, total and occupied	Stock table volume
Stock records, total	`/SCWM/AQUA` sizing
Peak factor and when	Seasonal, month-end, promotional
Operating hours	Whether a maintenance window exists at all

23.2 Peak is the design point

Warehouses are peak-driven businesses. A design sized for average throughput fails on the busiest day of the year, which is also the most expensive day to fail.

Ask specifically: what is the busiest hour of the busiest day, and what multiple of average is it? In retail distribution this is routinely 3–5x. Design and test against that number, not the average.

23.3 The non-functional requirements

Agree these in writing, with numbers, before design:

NFR	Typical target	Consequence if missed
RF screen response	Sub-second at the terminal	Operators wait; throughput collapses
Wave release duration	Inside the shipping window	Trucks leave late
Label print latency	Seconds from trigger	Operators wait at the packing bench
System availability	Often 99.5%+ for 24/7 sites	Warehouse stops
Recovery time objective	Minutes, not hours	Extended manual operation
Goods issue to ERP latency	Minutes	Customer billing delayed

RF response time is the one that decides whether the project is judged a success. Operators feel every extra half-second, thousands of times a shift. Measure it on the device over the warehouse WiFi, not in a desktop simulator on the office network.

23.4 Number ranges

An unglamorous architect responsibility that causes real incidents:

- Warehouse task, warehouse order, delivery, HU, TU, wave, physical inventory document — each has a range
- **SSCC ranges for shipping HUs** are externally meaningful and must not be reused; exhaustion is a customer-facing incident
- Size ranges against projected volume over the system’s expected life, not next year
- Document explicitly who monitors range consumption and at what threshold

23.5 The deliverable

A volumetrics and NFR document with the numbers, the peak factors, the agreed targets, the number range plan, and the source of each figure. When throughput is questioned after go-live, this document is what settles whether the design or the assumption was wrong.

Chapter 24: Authorisation, Security and Data Separation

24.1 The EWM authorisation dimensions

EWM authorisation is multi-dimensional in a way that ordinary ERP authorisation is not:

Dimension	Restricts
Warehouse number	Which warehouse a user may work in
Activity area	Which part of the warehouse
Warehouse process type	Which movements a user may perform
Party entitled to dispose	Whose stock a user may see — critical in 3PL
Storage type	Which zones
RF menu	Which transactions appear on the device
Work centre	Which stations a user may operate

24.2 The RF menu is an authorisation surface

The RF menu determines what an operator can even see on the scanner. It is both a usability and a security control, and it is frequently designed as usability only.

Design rule: menu structure follows role, and the roles follow the operational job — picker, receiver, packer, supervisor. An operator scrolling past nine irrelevant options loses seconds every transaction.

24.3 Third-party logistics data separation

In a 3PL warehouse, **party entitled to dispose is a contractual boundary, not a data field.** Client A must not see client B's stock, deliveries, or performance data.

Every custom object must respect it:

- Custom reports filtered by entitled party, always
- Custom RF transactions restricted by the operator's assigned client
- Custom interfaces scoped to one client's data
- Analytics and dashboards partitioned

Consequence of failure: a contractual incident with commercial consequences, not a defect ticket. Treat this as a design constraint with the same weight as a legal requirement, and test it explicitly with a restricted user.

24.4 Resources, users and the physical link

An EWM resource represents a person or a device and is bound to a user session during RF logon. This creates the link between a system action and a physical operator — which is what makes labour measurement and traceability possible.

Design questions: is a resource a person or a device? What happens at shift change? Can two operators share a device sequentially, and does the system know? These have consequences for labour management and for any traceability requirement.

24.5 The deliverable

An authorisation concept covering the dimensions above, the role catalogue mapped to operational jobs, the RF menu design per role, the 3PL separation rules if applicable, and the resource-to-user model. Reviewed by the client's security team, and tested with restricted users rather than assumed.

Chapter 25: UI, Monitoring and Analytics Strategy

25.1 The UI landscape

EWM has more distinct user interfaces than almost any SAP area, and choosing correctly per user group is an architect decision.

Interface	User	Use for
RF (device)	Operators	All physical execution
Work centre UI	Packers, deconsolidators, VAS operators	Station-based work
Warehouse management monitor	Supervisors, support	Visibility, exception handling, manual intervention
Fiori apps	Supervisors, managers, planners	Modern task-based work and analytics
Classic transactions	Configuration, support	Setup and deep troubleshooting

The rule: operators use RF, supervisors use the monitor and Fiori, and nobody in the warehouse uses a classic transaction. A design that puts a classic transaction in an operator's hands is a design that has not understood the environment.

25.2 The warehouse monitor as an architecture asset

The monitor is EWM's central visibility tool and it is extensible — you can add nodes, methods and hierarchies. Before building a custom report, ask whether it should be a monitor node instead.

Advantages of extending the monitor over a custom report: it inherits the existing navigation, authorisation, drill-down and follow-on actions, and supervisors already know how to use it.

This is the EWM-specific instance of the general rule: extend the standard tool rather than building beside it.

25.3 Alert and exception design

A warehouse runs on exceptions. The architect decides what surfaces, to whom, and how fast.

- Which conditions raise an alert — stuck task, blocked queue, replenishment shortfall, failed telegram, undistributed delivery
- Where it appears — monitor, Fiori, email, a screen on the floor
- Who owns each alert type and what the response is
- Escalation when nobody responds

The failure to avoid: alerts nobody owns. An alert list that is permanently long is an alert list nobody reads, and it hides the one that mattered.

25.4 Analytics and KPIs

The warehouse KPIs an architect should ensure are measurable from day one:

- Picks per operator hour, by activity area and shift
- Order fulfilment cycle time, receipt to dispatch
- Dock-to-stock time
- Inventory accuracy from cycle counting
- Wave release to shipment completion
- Task confirmation rate against plan
- Exception rate by type

Build these on CDS views over the EWM data model rather than extracting to spreadsheets. **Check the API State of any `/SCWM/` entity you consume** (Chapter 14.4) — an analytics layer built on unreleased objects is debt.

25.5 The deliverable

A UI and reporting strategy: interface per user group, RF menu structure, monitor extensions planned, alert catalogue with owners, KPI definitions with their data sources, and the analytics architecture.

Chapter 26: Business Continuity and Degraded Mode

The chapter most EWM designs omit entirely, and the one operations will ask about first.

26.1 The question that must have an answer

“What does the warehouse do when EWM is unavailable?”

In an ordinary ERP outage, users stop working and catch up. A warehouse cannot stop — trucks are at the dock, product is on the conveyor, and there are people on shift being paid. The answer “we have high availability” is not an answer; it is a risk mitigation.

26.2 Degraded mode design

Design for three tiers of failure, each with a defined procedure:

Failure	Duration	Warehouse response
RF network outage	Minutes	Operators continue on paper for pre-printed work; capture into the system on recovery
EWM unavailable	Hours	Full manual mode against pre-printed pick lists and a recorded log; bulk entry afterwards
ERP unavailable (decentralized)	Hours	Warehouse continues; queues build; catch-up on recovery

That third row is the entire argument for decentralized deployment (Chapter 1.1), and it is the concrete way to explain the trade-off to a sponsor.

26.3 What the design must specify

- **Pre-printed fallback documents** — what, how often, and who holds them
- **The manual recording method** during outage, and who is responsible
- **The catch-up procedure** — how manually executed work enters the system, and in what order
- **Stock integrity after catch-up** — expect a reconciliation, plan for it
- **The decision authority** — who declares degraded mode and who declares recovery
- **Communication** — to operators, to ERP users, to carriers

26.4 Planned downtime

Even a 24/7 warehouse needs upgrades. The design must answer:

- Is there any window at all, and how long?
- Can the warehouse run degraded during a planned window?
- In decentralized, can EWM stay up while ERP is patched? (This is a major practical advantage and worth stating explicitly.)
- What is the sequence for stopping and restarting MFS and the equipment?

26.5 The deliverable

A business continuity design covering the three failure tiers, the fallback documents, the catch-up procedure, the reconciliation approach, and the decision authority. Signed off by warehouse operations, because they are the ones who will execute it at three in the morning.

Chapter 27: Cutover, Stock Migration and Go-Live

Warehouse cutover is unlike any other SAP cutover, because the warehouse is physically full of stock that must be correct on the first morning.

27.1 Why it is harder

- Stock is physical and must match the system exactly, bin by bin
- The warehouse cannot stop for long — every hour of shutdown is unshipped orders
- Operators must learn new devices and new transactions overnight
- Automated equipment must be reconfigured and re-tested
- A wrong bin in the load means an operator sent to an empty location on day one

27.2 The stock migration approach

The sequence that works:

1. **Freeze physical movement** at a defined moment
2. **Full physical inventory** in the legacy system, or a verified count
3. **Extract stock** at bin, product, batch, HU and stock type granularity
4. **Transform** to the new bin naming and structure (Chapter 22.3)
5. **Load** into EWM — via the standard initial stock upload mechanism, not by direct table writes
6. **Verify** by reconciliation report and physical spot checks
7. **Release** the warehouse

The transformation step is where projects fail. Legacy bin identifiers rarely map cleanly to the new scheme, and stock characteristics (batch, stock type, owner) are often less granular in the legacy system than EWM requires. Discover the gaps in a trial load, not on cutover night.

27.3 Run at least two trial loads

Non-negotiable. A trial load reveals:

- Data quality gaps in the legacy extract
- Mapping failures in the transformation
- Load runtime against real volume — which determines the cutover window
- Reconciliation discrepancies and their causes

Time the load. If loading 200,000 stock records takes eleven hours and the window is six, you need to know that in a trial, not at 2am on cutover weekend.

27.4 Cutover strategy

Strategy	When	Risk
Big bang	Single warehouse, manageable volume	Highest — everything changes at once
By warehouse	Multi-site rollout	Lower; template discipline needed
By storage type or area	Large site, phased	Complex — two systems govern one building
Parallel run	Rarely feasible	Double data entry; usually impractical in a warehouse

Phased-by-area is attractive on paper and hard in practice, because stock moves between areas and the boundary needs a bridge process. Choose it only with a clear answer for cross-boundary movements.

27.5 The go-live period

- **Staff the floor, not the office.** Support presence at the dock and in the aisles on day one, not on a conference bridge.
- **Reduce volume deliberately** for the first days if the business can absorb it. Ask for it early; it is a commercial decision with lead time.
- **Extra supervisors**, since operators will be slower and will need help.
- **Daily reconciliation** between EWM and ERP stock during hypercare, not weekly.
- **A visible defect triage process**, because floor staff will report issues faster than any ticketing tool can absorb them.

27.6 The deliverable

A cutover plan with the stock migration approach, the trial load schedule and results, the timed runbook with go/no-go points, the rollback position, the hypercare staffing model, and the reconciliation procedure.

Define the rollback point explicitly. After stock is loaded and physical movement resumes, rollback is no longer a technical action — it is a physical recount. Everyone should know when that line is crossed.

Chapter 28: Migration Paths

The commercially valuable chapter, and the one that most needs verifying against a real estate. The frameworks here are sound; the tooling details are release-specific.

28.1 The two migrations

Legacy WM to EWM. A different product with a different data model, not an upgrade. Configuration, custom code and processes are all rebuilt. Stock is migrated. Users are retrained.

SCM EWM 9.x to S/4HANA EWM. Same product lineage, so configuration and much custom code carry forward — but the target deployment decision (embedded versus decentralized) reopens the architecture, and the ABAP stack changes underneath.

These are different projects with different risk profiles. Conflating them in a proposal is a credibility failure.

28.2 WM to EWM — the method

1. **Inventory the legacy estate.** WM configuration, custom transactions, custom tables, RF/ITSmobile transactions, interfaces, reports, forms.
2. **Map processes, not objects.** WM movement types do not map to EWM process types one-to-one. Start from what the warehouse actually does.
3. **Design the warehouse structure fresh** (Chapter 22). Do not transliterate the WM structure — WM storage type conventions frequently encode workarounds for WM limitations that EWM does not have.
4. **Triage the custom estate** exactly as in a clean core remediation: *retire* (no longer executed or superseded by EWM standard), *replace* (EWM standard covers it), *rebuild* (needed, no standard equivalent), *defer*. Usage data from the legacy system drives this.
5. **Rebuild, do not port.** WM custom code cannot be moved to EWM. Every rebuilt object is new development against the EWM framework.
6. Stock migration and cutover (Chapter 27).

The single most valuable observation to make to a WM client: a large share of their custom WM code exists to work around WM's limitations, and EWM does those things as standard. Storage control, deconsolidation, wave management, labour capture and HU handling all replace custom WM developments routinely. **Triage before you estimate a rebuild**, or you will quote for rebuilding things that no longer need to exist.

28.3 SCM EWM 9.x to S/4HANA EWM — the method

1. **Decide embedded or decentralized** (Chapter 1.1) before anything else. This is the architecture decision the migration reopens.
2. **Assess the custom code** against the S/4HANA ABAP stack — simplification items, removed objects, and the ABAP Cloud question (Chapter 14.4).
3. **Assess the configuration** for changes between releases.
4. **Assess the integration layer**, which changes materially if moving from decentralized to embedded — an entire interface layer disappears.
5. **Plan the MFS re-verification** if automated equipment is involved. Telegram handling must be re-tested against the equipment, not assumed.

6. Cutover, which is simpler than WM→EWM because the data model is compatible.

28.4 The decision that clients get wrong

Moving from decentralized SCM EWM to **embedded** S/4HANA EWM removes the integration layer, which is genuinely attractive — but it also couples the warehouse's availability to the ERP system's upgrade calendar. A 24/7 distribution centre that was independent becomes dependent.

Ask Chapter 1.1's question again during migration, because the answer may have changed since the original implementation, in either direction. Do not assume the existing deployment model is still correct.

28.5 Where a technical specialist adds the most value

In both migrations, the scarce skill is the same: **reading a large legacy custom estate, deciding what should exist in the target, and rebuilding it in the new framework compliantly.** That is triage plus framework knowledge, and it is precisely the intersection of this manual and the clean core method.

Very few people can do both halves. That is the position worth occupying.

28.6 The deliverable

A migration assessment: legacy estate inventory with usage data, process map, custom object disposition register, target warehouse structure, deployment model recommendation with the availability rationale, effort model by disposition class, phased roadmap, and the cutover approach.

This is the same shape as the clean core assessment in the companion manual. The method transfers; only the domain changes.

Part IX — Worked Examples

Chapter 29: Custom Putaway Determination

29.1 The requirement

> *For hazardous products, putaway must go only to bins in designated hazmat storage sections, and only if the section's current hazmat volume plus the incoming quantity stays under a legal limit held per section.*

29.2 Working the extension hierarchy (Chapter 14.1)

1. **Configuration?** Storage type search and bin determination handle section restriction by product characteristics. The **capacity limit against a legal volume threshold** is not standard. Partial fit.
2. **Packaging specification?** Not applicable.
3. **PPF?** No — this is determination, not output.
4. **BAdI?** Yes. A bin determination BAdI, filtered to hazmat process types.

Verdict: configuration for the section restriction, BAdI for the volume check. Note that this is a *split* answer. Requirements are rarely purely one or the other, and proposing the split rather than coding all of it is what an architect does.

29.3 The design

```
" Thin BAdI implementation — logic in a testable class
METHOD if_ex_scwm_bin_determination~change_bins.   " [VERIFY]
  CHECK is_context-procty IN gr_hazmat_process_types.

  DATA(lo_hazmat) = NEW zcl_ewm_hazmat_capacity(
                          iv_lgnum = is_context-lgnum ).

  ct_bins = lo_hazmat->filter_by_capacity(
              it_bins     = ct_bins
              iv_matid    = is_context-matid
              iv_quantity = is_context-quan ).
ENDMETHOD.
```

```
CLASS zcl_ewm_hazmat_capacity IMPLEMENTATION.
  METHOD filter_by_capacity.
    " 1. One set read of current hazmat volume per candidate section
    "    – NOT a read per bin (Chapter 20.2)
    DATA(lt_section_volume) = get_current_volume(
                                VALUE #( FOR b IN it_bins ( b-lgber ) ) ).

    " 2. One read of the configured limits
    DATA(lt_limit) = get_section_limits( ).

    " 3. Filter in memory using hashed lookups
    LOOP AT it_bins ASSIGNING FIELD-SYMBOL(<bin>).
      READ TABLE lt_section_volume WITH TABLE KEY lgber = <bin>-lgber
           ASSIGNING FIELD-SYMBOL(<vol>).
      ...
    ENDLOOP.
  ENDMETHOD.
ENDCLASS.
```

29.4 The decisions that matter

1. **Split configuration and code**, rather than coding both halves.
2. **Filter early in the BAdI** — `CHECK` on process type, so non-hazmat putaway pays almost nothing. This BAdI fires on every putaway in the warehouse.
3. **Set reads, not per-bin reads.** Bin determination can be handed hundreds of candidates.
4. **Logic in a class**, so it is unit testable without a warehouse.
5. **The limits are configuration data, not constants** — a legal threshold will change and must not require a transport.

29.5 Testing

Unit tests on `zcl_ewm_hazmat_capacity` with doubled volume and limit data: under limit, exactly at limit, over limit, no limit configured, no candidate bins remaining. That last case matters — what should happen when every bin is rejected? Decide it deliberately: error, or fall back to a default section.

Chapter 30: A Custom RF Transaction

30.1 The requirement

> *Operators must record a quality sample during goods receipt: scan the HU, scan or enter the sample quantity, select a reason code, and trigger a follow-on quality inspection.*

30.2 The first question (Chapter 16.6)

Could a standard transaction plus a BAdI do this? Investigate goods receipt RF transactions and their extension points. Only if none can carry the extra fields does a custom transaction become justified. Record the investigation — an architect who builds

a custom RF transaction without documenting why is creating a permanent liability with no rationale attached.

Assume here that no standard step accommodates the reason code.

30.3 The design

Steps: scan HU → enter quantity → select reason → confirm.

Four steps means four network round trips. **Design decision:** combine quantity and reason onto one screen where the device size permits, reducing to three. On a shift with 400 samples that is 400 fewer round trips.

Per step:

Step	Validation	Failure behaviour
Scan HU	HU exists, in this warehouse, in GR status	Message, stay, re-focus the scan field
Quantity + reason	Quantity ≤ HU quantity; reason valid	Message, stay, keep entered values
Confirm	Full consistency re-check	Message, return to step 1

Keeping entered values on error is a small detail that operators notice every shift.

30.4 The code shape

Each step handler: read the screen fields, validate, delegate to `zcl_ewm_qs_sample_service`, set the next function code. No business logic in the handler (Chapter 16.5).

The service class owns: HU validation, quantity checks, creating the sample record, and triggering the inspection. It is fully unit testable with doubles for HU reads and the inspection API.

30.5 What to test

- Every error path on every step
- Both device profiles in use
- Network drop between step 2 and step 3 — what state is the HU left in?
- Concurrent sampling of the same HU by two operators
- Volume: 400 samples in a shift, measured seconds per sample

30.6 The architecture record

Write it down: why a custom transaction rather than standard plus BAdI, which standard transactions were assessed, what the maintenance implication is, and under what condition it would be retired.

Part X — The Architect Path

Chapter 31: What to Learn, In What Order

31.1 The sequence

Depth first, in this order. Each stage assumes the one before.

Stage 1 — Orientation (weeks 1–4). Run the processes yourself in a system: inbound, outbound, physical inventory. SQL trace each one (Chapter 2.4). Learn the monitor. Learn the data model by observation, not by reading table lists.

Stage 2 — The process objects (weeks 5–12). Warehouse task and order, storage control, warehouse process types, WOCR, packaging specifications, handling units. Configure them yourself. **You cannot be a technical architect in EWM without configuration literacy** — the boundary between config and code is where every design decision sits.

Stage 3 — The extension framework (months 4–6). BAdI landscape, PPF, the extension hierarchy. Build something small.

Stage 4 — RF (months 6–9). The highest-differentiation skill after MFS. Build a custom transaction end to end, including the error paths.

Stage 5 — Integration and architecture (months 9–12). Decentralized integration design, monitoring, master data replication, the availability trade-off. This is where the architect role actually lives.

Stage 6 — MFS (optional, later). A separate specialisation decision (Chapter 17.5).

31.2 What you cannot learn without a system

All of it. EWM is not learnable from documentation — the behaviour is emergent from configuration and you have to see it. Your options, best first:

1. **A client project.** Any role. This is worth more than everything else combined.
2. **An employer sandbox** with a configured warehouse.
3. **SAP Learning Hub with a practice system**, if it includes EWM for your scenario.
4. **A cloud-hosted appliance**, if your employer will fund the infrastructure.

Given your hardware, nothing runs locally. **Solve system access before buying any course.** This is the step that stalls people, and it is the same failure mode as the Docker install earlier — the learning plan dies at the environment, not at the material.

31.3 The honest timeline

- **Months 1–6:** you are a developer who can work in EWM
- **Months 6–12:** you are an EWM developer
- **Months 12–24:** with one full implementation or migration delivered, you are credible as a technical architect

- **Two or more projects, including one migration:** you are one

Nobody becomes an EWM solution architect from a manual. The manual shortens stage 1 and 2 and gives you the vocabulary to be useful sooner. Delivery does the rest.

31.4 What differentiates you specifically

You bring RAP, CDS, clean core and Fiori depth that most EWM developers do not have. The intersection worth owning:

Clean-core-compliant EWM extensibility, and the technical rebuild of custom WM/EWM code during migration.

Every WM-to-EWM and EWM 9.x-to-S/4 migration raises the question “how do we rebuild these custom objects without wrecking the core,” and almost nobody can answer it. That question needs both the EWM knowledge in this manual and the release-contract knowledge you already have.

31.5 The trap

Not learning EWM. **Starting EWM and abandoning it at month four.** The value in this path is entirely back-loaded — nothing pays until you have delivered something. Half a specialisation is worth nothing, and it costs the same twelve months as the alternative you did not pursue.

Decide now whether you are doing this or the clean core path. Doing both means doing neither.

Appendix A: The EWM Diagnostic Sequence

When something behaves unexpectedly, in this order:

1. **Look at the object in the warehouse monitor** before debugging anything
2. Which warehouse process type, and what does it configure?
3. Which storage control applied — process-oriented, layout-oriented, both?
4. What did packaging specification determination produce?
5. Which WOCR was selected, and what are its limits?
6. Which BAdI implementations are active on this path?
7. Is the task open (`ORDIM_O`) or confirmed (`ORDIM_C`)?
8. Is this a queue or replication problem rather than a warehouse problem?
9. Only now: debug

Appendix B: Design Review Checklist

- ☐ Deployment model justified by the availability requirement
- ☐ Extension hierarchy worked down, rejected options recorded
- ☐ Configuration exhausted before code
- ☐ BAdI confirmed to fire on every path — RF, background, API
- ☐ Logic in a testable class, hooks kept thin
- ☐ Stock reads restricted
- ☐ No database read inside a task or item loop
- ☐ RF round trips minimised, error paths designed
- ☐ Exception codes handled
- ☐ Custom field propagation estimated across all boundaries
- ☐ Queue monitoring designed (decentralized)
- ☐ Stock reconciliation designed (decentralized)
- ☐ MFS: timeouts, logging, PLC restart procedure
- ☐ Tested at production volume
- ☐ Tested on every device profile
- ☐ API State checked on every `/SCWM/` object consumed

Appendix C: Architect Deliverable Checklist

- ☐ Warehouse structure document, signed off by operations
- ☐ Bin naming convention with extension allowance and sort verification
- ☐ Activity area design with the work-organisation rationale
- ☐ Volumetrics collected as numbers, including the peak factor
- ☐ NFR targets agreed in writing, RF response time included
- ☐ Number range plan sized over system life, with a monitoring owner
- ☐ Authorisation concept covering all EWM dimensions
- ☐ 3PL data separation tested with a restricted user (if applicable)
- ☐ UI per user group decided; RF menu structured by operational role
- ☐ Alert catalogue with a named owner per alert type
- ☐ KPI definitions with data sources
- ☐ Degraded mode procedure for all three failure tiers
- ☐ Planned downtime approach agreed with operations
- ☐ Stock migration approach with at least two trial loads scheduled
- ☐ Cutover runbook with timed steps and an explicit rollback point
- ☐ Hypercare staffing on the floor, not on a bridge

Appendix D: Object Verification Log

Keep this as you learn. It replaces the object names this manual could not verify for you.

Purpose	Object name	Type	Release	Released?	Verified on
Create warehouse task		FM			
Confirm warehouse task		FM			
Read delivery		Class			
Read stock		FM			
Read HU		FM			
Bin determination		BAdI			
WOCR influence		BAdI			
Storage control		BAdI			
RF step contract		—			

Fill this in during your first month with system access. It will be more valuable to you than any chapter here, because it is verified.

Closing Note

The structure, the decision frameworks, the diagnostic sequences and the design rules in this manual are stable and correct. The object names are not verified and must not be treated as though they were.

The most important pages are Chapter 2.4 — learning the data model by SQL trace rather than by reading — and Appendix D, which is empty and is meant to be filled in by you against a real system.

EWM cannot be learned from a document. It can be learned much faster with one, provided the document is honest about which parts it cannot guarantee.