MN7R PRODUCT GUIDE

THE COMMODITY BROKERAGE WORKSPACE FOR MODERN TEAMS

MN7R

PRODUCT GUIDE

The Commodity Brokerage Workspace for Modern Teams

Deals. Clients. Execution. Intelligence.
All in **One** Place.

TRANSPARENCY • CONTROL • COLLABORATION • EXECUTION • RESULTS

DEALS TO TRADES

CLIENT RELATIONSHIPS

EXECUTION EXCELLENCE

AI-POWERED INSIGHTS

SECURE & GOVERNED

BUILT FOR BROKERS. DESIGNED FOR PERFORMANCE.

All product descriptions reflect the current public-safe state of MN7R at the time of generation.

No real client, broker, trade, payment, banking, pricing, or private workspace data is included.

This guide is not the internal MN7R operating manual. The internal handbook remains the role-specific reference for onboarding, support, permissions, and detailed workflow training.

PUBLISHED BY MN7R

MN7R Product Guide

The Commodity Brokerage Workspace for Modern Teams

Last updated: June 16, 2026

Generated: June 16, 2026

Version 1.2.0 · Revision 6b001e4

Published by MN7R

Table of Contents

How to Read This Guide

MN7R is a working product, not a theoretical software concept. This guide should therefore be read as a practical field guide to the current public-safe product, not as a marketing brochure and not as a technical specification.

The purpose of the guide is simple: to help a broker, team leader, operations manager, partner, or potential client understand what MN7R is built to do.

The guide follows the natural order of commodity brokerage work.

First, it explains why brokerage teams need a dedicated workspace instead of relying only on chats, spreadsheets, phone calls, and individual memory.

Then it walks through the commercial workflow: offers, bids, matches, and trades.

After that, it explains the supporting layers: clients, execution, templates, analytics, roles, and AI-assisted operating support.

Finally, it explains the boundaries: what MN7R helps automate, what remains under human review, and what the system does not do automatically.

This distinction matters. In physical commodity brokerage, software must help people move faster, but it must not silently take over commercial, legal, payment, or contractual decisions. MN7R is designed around that principle.

Chapter 1. What MN7R Is

MN7R is a commodity brokerage workspace for teams that need one structured place to manage market flow, client memory, trade execution, analytics, and supervised AI-assisted operating work.

That sentence sounds simple, but it describes a very specific problem.

Commodity brokerage does not behave like ordinary sales. A broker is not only managing a list of leads. A broker is constantly working with physical goods, origins, destinations, delivery periods, basis terms, transport routes, changing prices, seller interest, buyer interest, counterparty preferences, payment terms, execution risks, and follow-up obligations.

A seller may offer corn FCA Romania for June shipment. A buyer may be looking for corn DAP Northern Italy in the same period. A broker sees the possible relationship between the two, but the deal is not done yet. There may be a price gap. Freight may make it workable or impossible. The buyer may accept a different origin. The seller may change the validity. The same idea may be reposted tomorrow. A counter may appear. The match may become a trade. After the trade, execution begins, and the work continues through confirmations, contracts, delivery, invoices, payments, and commission.

In many teams, this entire chain is scattered across chats, spreadsheets, personal notes, screenshots, inboxes, and memory.

MN7R is built to give that chain a structure.

The core idea is that a market idea should not disappear after it has been posted, forwarded, discussed, or copied into a message. It should remain connected to its commercial context: who offered it, who wanted it, what commodity it concerned, what basis was used, what delivery place was named, which broker worked it, what happened later, whether it became a match, whether it became a trade, and whether that trade later moved into execution.

This is why MN7R is not only a CRM and not only a deal board.

It is a workspace for the full brokerage path from market interest to execution follow-up.

The product is built around several connected areas.

Deals is the daily market workspace. It is where brokers create and work with offers, bids, matches, and trades.

Clients is the company and relationship layer. It helps the team keep client identity, broker ownership, contacts, notes, legal context, logistics context, and follow-up structure more consistent.

EXE is the execution workspace. It is where completed trades become contract follow-up: confirmations, delivery balance, invoice status, payment status, assigned officer, support notes, drafts, templates, and execution pressure.

Secondary Views and analytics give the team market and operational context: prices, index references, liquidity, reports, and other supporting views that help brokers and managers understand where attention is needed.

The AI Officer Assistant is the supervised AI layer built primarily around execution. It helps detect missing facts, organize findings into action items, prepare reviewed drafts, follow playbook stages, maintain timeline memory, and keep execution risks visible. It does not replace human responsibility. It is designed to support the human officer and supervisor, not silently make legal or financial decisions.

This role-aware structure is central to MN7R.

A broker needs speed. A boss needs team visibility. An admin needs control and governance. An execution officer needs operational discipline after the trade. MN7R gives each role a different working angle on the same underlying flow.

A broker may use MN7R to post and update a bid, compare it with offers, create a trade, and follow execution on deals where they participated.

A boss may use MN7R to see whether the team is actually progressing useful opportunities, whether clients are assigned correctly, whether stale records are creating noise, and whether execution pressure is building.

An admin may use MN7R to manage reference data, review role behavior, inspect lower-role experiences, support onboarding, and keep controlled dictionaries and governance flows from drifting.

An execution officer may use MN7R to keep completed trades from becoming a mess of disconnected emails and messages.

That is the central product promise of MN7R: not generic collaboration, but structured brokerage continuity.

It connects the market side of the business with the operational side of the business.

It lets a broker team move from:

“We talked about this somewhere”

to:

“This is the offer, this is the bid, this is the match, this is the trade, this is the execution state, and this is what needs to happen next.”

Chapter 2. Why Commodity Brokerage Needs a Workspace

Commodity brokerage teams often begin with the fastest tools available.

Telegram groups. WhatsApp chats. Phone calls. Excel files. Google Sheets. Email threads. Screenshots. Personal notebooks. A few saved templates. A folder of contracts. A manager asking, “Where are we on this one?”

This is natural. Brokerage is fast. The market moves quickly. A broker cannot stop every conversation to fill out a perfect enterprise system.

But over time, the same speed creates a hidden cost.

A price is discussed in one chat, but the validity is written somewhere else. A seller confirms a quantity, but the broker forgets whether it was for the first half of the month or the full month. A buyer asks for an alternative delivery place, but the note stays in a private message. A position is reposted as fresh, but the older version remains in circulation. A match looks promising, but no one follows it. A trade is concluded, but execution begins with missing information. A payment is delayed, but the team does not see the pressure clearly until it becomes urgent.

The problem is not that brokers are disorganized people.

The problem is that commodity brokerage creates more context than ordinary communication tools can safely hold.

A single physical commodity idea can contain many moving parts:

the commodity;

the crop year;

the origin;

the quantity;

the tolerance;

the basis;

the delivery place;

the transport mode;

the delivery period;

the price;

the currency;

the VAT treatment;

the payment terms;

the seller or buyer;

the broker owner;

the counterparty preferences;

the matching logic;

the later trade;

the execution state.

When this context lives only in messages and spreadsheets, three things happen.

First, the team loses continuity.

A market idea may be visible today but difficult to reconstruct next week. The broker may remember the conversation, but the system does not. If that broker is absent, the team is left with fragments.

Second, the team loses comparability.

If one broker writes “DAP Italy”, another writes “Northern Italy”, a third writes a specific destination, and a fourth stores the route in a note, then matching, analytics, and reporting become inconsistent. The team may still work, but the system cannot reliably compare opportunities.

Third, the team loses control after the deal.

A trade is not the end of the work. In physical commodity brokerage, the trade is the start of the execution tail. After commercial agreement, the team still needs confirmations, contracts, instructions, delivery documents, invoices, payment tracking, commission follow-up, and exception handling.

If execution is not connected to the original commercial record, the officer must reconstruct the deal from messages.

This is where errors appear.

Was the buyer name correct?

Which broker worked the seller side?

Was the delivery period 1–15 June or full June?

Did the buyer send instructions?

Was the unload weight confirmed?

Is the invoice pending?

Is payment overdue?

Can commission be reviewed?

These are not abstract administrative questions. They affect whether a contract is handled smoothly or becomes a problem.

MN7R exists because commodity brokerage needs a workspace that respects the speed of the market while preserving the structure of the work.

The product does not try to replace the broker's judgement. It does not try to force every market conversation into a rigid enterprise form before the conversation can happen.

Instead, it gives the team a practical operating layer.

Offers and bids can be created quickly, but they remain structured enough to be matched, filtered, updated, reposted, reviewed, and carried forward.

Matches can help brokers see where buyer and seller interest may overlap, but the broker still decides whether the opportunity is worth pursuing.

Trades can capture the actual commercial closure point, including seller and buyer, brokers on both sides, commodity, basis, route, volume, price, and commission context.

Clients can preserve company identity, relationship ownership, contacts, and operational context instead of letting every name become a new spelling in a spreadsheet.

EXE can turn completed trades into visible execution work, with confirmations, delivery, invoices, payments, officer assignment, and AI-assisted follow-up.

The key word is continuity.

MN7R is designed so that one piece of work can feed the next.

An offer can become part of matching.

A match can become a trade.

A trade can feed EXE.

EXE can create action items, drafts, timeline events, and review states.

The team can then see what is happening not only at the moment of posting, but across the whole operating life of a deal.

This is especially important for teams with multiple brokers, supervisors, and execution officers.

A single independent broker can survive for a long time on memory and messaging. A team cannot scale that way. Once several people work the same market, the same clients, and the same execution flow, the workspace becomes the memory of the business.

That is why MN7R is not simply a “nice interface”.

It is an attempt to turn brokerage from a collection of conversations into an operating system.

Chapter 3. The Core Workflow: From Offers and Bids to Trades

The core MN7R workflow follows the natural logic of brokerage:

Offers.

Bids.

Matches.

Trades.

Execution.

This is the basic route from market interest to operational follow-up.

An offer is seller-side market interest. It answers the question: what is someone willing to sell, under what terms, in which place, in which period, and at what price or price idea?

A bid is buyer-side market interest. It answers the question: what does someone want to buy, under what terms, in which destination or basis, in which period, and at what price or price idea?

A match is not a deal. It is a comparison between the seller side and the buyer side. It helps the broker see whether two pieces of market interest may be commercially workable.

A trade is the commercial closure point. It records what was actually agreed.

EXE is what happens after the trade. It is where the commercial agreement becomes execution work.

This structure may sound obvious, but many brokerage teams do not actually work with this continuity. They may have a list of offers, a separate list of bids, informal match discussions, a trade file, and a separate execution document. The handoffs are manual. The context is carried by people.

MN7R keeps these stages connected.

In Deals, brokers create and manage offers and bids. The forms are built around practical commodity brokerage fields: commodity, origin, crop year, quantity, tolerance, basis, destination or delivery place, transport, delivery period, price, currency, payment terms where relevant, VAT markers, notes, counterparty, and broker identity.

Each record has a public reference identifier so the team can discuss a position without relying only on screenshots or copied messages. The identifier matters because brokerage

conversations move quickly. A broker can say, “Look at this offer,” and the team can return to a specific structured record.

The record can be updated. It can be reposted as fresh. It can be used in matching. It can become the source for a counter idea. It can eventually support trade creation.

This is different from a chat message.

A chat message may be fast, but it is not a durable operating object. A structured offer or bid can continue working after the first message was sent.

Matching is the bridge between market flow and commercial action.

MN7R can compare buyer and seller interest by commodity, route, basis, delivery place, delivery period, quantity, and price relationship. It can help the broker see that two sides may be close enough to justify attention. It can also show why a match is weak: wrong basis, wrong period, incompatible delivery place, unrealistic price gap, or missing quantity.

But MN7R does not pretend that matching is a fully automatic decision.

Physical commodity brokerage still depends on judgement. Freight can change the economics. A client may accept an alternative delivery place. A seller may adjust the price. A buyer may care about quality or timing more than the visible number suggests. A broker may know that a “weak” match is worth trying because of relationship context.

The match is therefore a working signal, not a final answer.

If the broker and counterparties agree, the opportunity becomes a trade.

The trade record is important because it is the handoff from market conversation to execution. It should capture the actual commercial agreement, not just the idea that started the conversation. This includes the seller, buyer, seller-side broker, buyer-side broker, commodity, basis, delivery place, transport, delivery period, volume, price, currency, payment terms, and commission context where known.

Once a trade is completed, it can feed EXE.

This is the point where many systems become weak. They treat execution as something separate from the market workflow. A trade is “done”, and then someone starts a new process in documents, emails, and messages.

MN7R treats the trade as the start of the next workflow.

In EXE, the team can see the completed trade as an execution record. The contract can be followed through confirmations, delivery balance, invoice state, payment state, officer assignment, notes, drafts, templates, action items, playbooks, timeline events, and AI-assisted findings.

This creates a clean chain:

a seller idea;

a buyer idea;

a comparison;

an agreement;

an execution process.

That chain is the backbone of the product.

It also explains why MN7R is role-aware.

A broker works mostly at the front of the chain: offers, bids, matches, trades, and client follow-up.

A boss looks across the team: whether brokers are creating useful flow, whether opportunities are being progressed, whether clients are covered, whether execution pressure is growing.

An admin supports the system: reference data, role behavior, governance, quality assurance, onboarding, and review.

An EXE officer works after the trade: confirmations, delivery, invoices, payment, commission, exceptions, and execution discipline.

The same underlying record can matter to all of these roles, but each role sees it from a different angle.

That is the reason for MN7R's structure. It is not only a database of positions. It is a workflow map for a brokerage team.

If the team works only in chats, the workflow is hidden inside conversations.

If the team works in MN7R, the workflow becomes visible:

What is on the market?

What might match?

What became a trade?

What is in execution?

What is blocked?

What needs review?

What should happen next?

For a brokerage team, those are the questions that matter every day.

Chapter 4. Offers

An offer is seller-side market interest.

In practical brokerage language, it is the structured record of what a seller is prepared to sell, under which commercial assumptions, and through which route, period, basis and price logic.

This sounds simple until you look at how offers actually appear in daily work.

A seller may offer corn from Romania for June shipment. Another seller may offer wheat from Ukraine on a port basis. A third may offer several oilseed products at once, with different qualities, different price levels, and different availability windows. Sometimes the price is firm. Sometimes the price is only an indication. Sometimes the quantity is fixed. Sometimes it is approximate. Sometimes the seller is open to several destinations. Sometimes the basis or transport changes the commercial meaning completely.

A generic CRM cannot understand this. A chat message can describe it, but it cannot keep it organized for future matching, reporting, reposting, supervision and execution.

MN7R treats an offer as a working market object.

The offer record can hold the seller, commodity, origin, crop year, volume, tolerance, basis, delivery place, transport, delivery period, price, currency, VAT context where relevant, payment terms where relevant, notes, broker ownership and status. It also gives the idea a visible public reference so the team can discuss it without relying only on screenshots or copied text.

This matters because an offer is rarely useful only once.

A seller-side idea may be posted today, updated tomorrow, reused next week, countered by a buyer, matched against several bids, or eventually converted into a trade. If the offer only exists as a message, the later workflow depends on human memory. If the offer exists as a structured record, the system can continue to use it.

In MN7R, offers live inside the Deals workspace. For the broker, this is one of the main daily operating surfaces. The broker can create, edit, repost, counter and work offer records. A boss can review team offers, spot stale or incorrect flow, and understand whether brokers are maintaining a useful seller-side market picture. An admin can inspect the same flow in a support, QA or onboarding context.

The important point is not only who can see the offer. The important point is what happens after the offer is saved.

Once an offer exists in MN7R, it can feed several downstream workflows.

It can appear in the Offers list.

It can become available for matching.

It can be used in counter flow.

It can support trade creation.

It can contribute to market analytics.

It can later become part of EXE follow-up if it ends in a completed trade.

This is the difference between a message and an operating record.

MN7R also supports the way real brokerage ideas often behave. A broker may repost the same commercial idea as fresh when the seller confirms it is still alive. A broker may work with multi-place structures when several delivery destinations are possible. A broker may use multi-crop or combo structures when a seller proposal includes several commodities or specifications under one visible commercial idea.

The goal is not to make the form complicated. The goal is to avoid flattening real market work into a single rigid row.

Another practical feature is publication support. Offers can be prepared for Telegram or other external relay formats, but MN7R treats publication as an output, not the source of truth. The structured offer remains in the monitor. The publication preview helps the broker produce a cleaner market message, but the working record stays available for future search, update, matching and review.

This distinction is critical.

If Telegram is the only place where an offer exists, the offer disappears into the flow of messages. If Telegram is only an output from MN7R, the team keeps both speed and structure.

For a broker, the offer workflow answers a daily question:

“What do I have to sell, under what terms, and what can I do with it next?”

For a boss, it answers a different question:

“Is the team keeping seller-side market flow fresh, accurate and commercially useful?”

For an admin or support user, it answers another:

“Is the workflow behaving consistently, and are the visible records using the right reference data, client names, statuses and publication logic?”

A good offer record is not paperwork. It is the starting point of market memory.

It keeps a seller-side idea alive long enough to become searchable, comparable, matchable, reportable and executable.

Chapter 5. Bids

A bid is buyer-side market interest.

It describes what a buyer wants to purchase, where the buyer needs it, under which basis, in which period, at which price or price idea, and with which commercial assumptions.

In daily work, a bid is not always as clean as a formal procurement order.

A buyer may say, “Show me corn DAP Northern Italy for June.” Another may ask for soybean meal CIF to a port, but only at a certain level. Another may want wheat, but without a final quantity yet. A processor may be watching several origins. A trader may want to see whether the market can reach a level before confirming volume.

MN7R is built to hold that practical uncertainty.

A bid can include the buyer, commodity, origin where relevant, crop year, quantity or open quantity, tolerance, destination, delivery place, basis, transport, delivery period, price, currency, VAT context where relevant, payment terms where relevant, notes, broker ownership and status.

Like offers, bids are not only stored for reference. They become active operating objects.

A bid can be updated.

It can be reposted.

It can be matched against offers.

It can be used in counter flow.

It can become part of a trade.

It can later support EXE follow-up after the trade is completed.

This is important because buyer interest often drives the urgency of the brokerage day.

A buyer may have a narrow delivery period. A buyer may need a specific origin. A buyer may be price-sensitive but flexible on route. A buyer may be looking for volume but not yet ready to declare final size. If this context sits only in a chat, the team may lose the nuance. If it sits in a structured bid record, it can be filtered, compared, reviewed and worked.

In MN7R, the bid workflow mirrors the seller-side offer workflow but with buyer-side logic.

For the broker, it is the main surface for recording and working buyer demand. For the boss, it provides visibility into whether the team is capturing real demand or only reacting to messages. For the admin, it helps support QA, role review and data consistency.

The bid record also helps prevent a common team problem: invisible demand.

In many brokerage teams, buyer interest is scattered across private chats. One broker knows a buyer wants corn. Another broker has a seller offer that might work. A third person remembers a route that could make the price possible. But unless the bid is visible as a structured record, the team depends on coincidence.

MN7R makes buyer-side interest part of the shared working market.

That does not mean every broker sees everything in every deployment. Visibility still depends on role, team and configured permissions. But the product model is designed around the idea that buyer demand should be more than a temporary message.

Bids also support more realistic pricing work.

A buyer can have a fixed price target, a price indication, or no final price yet. The bid can exist even when some pieces are still developing. This matters because brokerage often begins before everything is final. The system should help the broker work the interest, not reject the record because it is not yet a perfect contract.

A bid is also part of publication and report flow. A clean bid record can be turned into a structured market message. It can be included in internal review. It can support reports. It can feed analytics.

But again, publication is not the core. The record is the core.

The broker's practical question is:

"What do my buyers need, under which terms, and what seller-side opportunities could answer that need?"

The boss's question is:

"Where is buyer demand concentrated, and are brokers progressing it correctly?"

The admin's question is:

"Is the data structured enough to support matching, reporting, execution and governance?"

A bid is therefore not simply the opposite of an offer.

It is the demand-side memory of the team.

Chapter 6. Matches

A match is where MN7R helps the team see possible commercial overlap between seller-side and buyer-side interest.

It is important to define what a match is not.

A match is not a deal.

A match is not a contract.

A match is not an instruction to trade.

A match is not a replacement for broker judgement.

A match is a comparison layer.

It helps the broker see whether an offer and a bid may be worth attention.

In a physical commodity market, compatibility is not only about commodity and price. It may depend on origin, delivery place, basis, transport, delivery period, quantity, tolerance, quality, payment terms, route alternatives and commercial relationship context.

An offer may say FCA Romania. A bid may say DAP Northern Italy. The prices may look far apart until freight is considered. Or they may look close until logistics make the route unrealistic.

A seller may offer several ports. A buyer may name one destination but accept another. A note may include an alternate basis or place. Two records may not match perfectly on paper, but a broker may know they are close enough to try.

This is why matching in MN7R is designed as support, not automation.

The system can surface candidate overlaps. It can show that buyer and seller interest is aligned enough to be reviewed. It can help compare sides and keep the possible opportunity visible. But the broker still decides whether the opportunity is commercially serious.

A useful match can lead to several outcomes.

It may become a direct trade.

It may start a negotiation.

It may lead to a counter idea.

It may be rejected.

It may be parked for later follow-up.

This flexibility is important because commodity brokerage is not a simple marketplace where every compatible row instantly becomes a transaction. The match is a signal that the broker should look closer.

MN7R supports matching because broker teams often miss opportunities when the market flow is too scattered.

Imagine a buyer-side broker has a bid for corn. A seller-side broker has an offer that may work after freight. Both records exist, but if the two brokers are busy, the opportunity may sit unnoticed. Or one side may be in an old message, while the other is in a spreadsheet. Or the match may be visible only to the person who happens to remember both conversations.

Structured matching reduces that dependence on memory.

It also helps managers.

A boss can look at the match flow and ask practical questions:

Are brokers progressing useful opportunities?

Are matches being ignored?

Are price gaps too wide?

Are certain commodities or routes close to trade?

Are brokers creating trades directly without using visible match logic?

Are there repeated near-matches that suggest a market corridor is becoming active?

For support and admin users, match behavior also helps reveal data quality problems. If matching repeatedly fails because delivery places, commodity labels or basis values are inconsistent, the problem may not be the market. It may be reference data or record structure.

This is another reason MN7R connects matching with controlled dictionaries, clients, offers, bids and public IDs. Matching quality depends on data quality.

But it should never become blind automation.

A good match asks a question:

“Is this opportunity worth working?”

It does not answer:

“Do this trade.”

The human broker remains central.

MN7R’s role is to make the possible opportunity visible, structured and easier to evaluate.

Chapter 7. Trades

A trade is the commercial closure point in MN7R.

It is the moment when market work turns into a record of what was actually agreed.

This is a different kind of object from an offer, bid or match.

An offer may be an idea. A bid may be a demand signal. A match may be a possible opportunity. A trade is the point where the team says: this deal happened, these were the parties, these were the terms, and this record now needs to support execution.

A trade can be created directly or from a match. It may include seller, buyer, seller-side broker, buyer-side broker, commodity, basis, delivery place, transport, delivery period, volume, tolerance, price, currency, VAT context, payment terms, notes and commission details where known.

Seller-side and buyer-side brokers matter because brokerage is often collaborative. One broker may bring the seller. Another may bring the buyer. Sometimes the same broker works both sides. Sometimes the relationship history matters later for commission, follow-up and reporting.

MN7R makes those roles explicit.

Commission details may also be captured when known. They should not block every trade creation case, because real brokerage work can close before every downstream administrative detail is final. But when commission amount, currency, VAT mode or timing is known, the trade can preserve that context for later execution and accounting follow-up.

This is where the difference between a casual deal note and a structured trade record becomes visible.

If the trade is only written in a message, the execution team may later ask:

Who was the seller?

Who was the buyer?

Which broker worked which side?

What was the exact basis?

Which delivery period was agreed?

Was the price VAT-inclusive?

Was commission agreed on one side or both?

Was payment timing known?

Which note was final?

If those answers are already in the trade record, execution begins with less friction.

MN7R treats completed trades as the handoff into EXE.

This is one of the most important transitions in the product.

A deal is not truly finished when the commercial terms are agreed. In physical commodity brokerage, that is when a new operational process begins. Contracts must be confirmed. Documents may need to be prepared. Signatures may be tracked. Delivery instructions may be needed. Shipment and unload data may appear. Invoices and payments must be followed. Commission may need review.

EXE exists because completed trades create execution work.

The better the trade record, the stronger the EXE workflow.

This is why MN7R connects Trades with Contract View. A completed trade can become part of the execution register. The team can then track confirmations, delivery balance, payment status, financial balance, assigned EXE officer, support notes, drafts, action items, timeline memory and AI-assisted findings.

The handoff matters not only for the execution officer, but for the whole team.

The broker can still follow execution on deals where they participated.

The boss can supervise whether completed trades are moving properly through execution.

The admin can review the process, inspect role behavior, support onboarding and help correct workflow issues.

The AI Officer Assistant can detect missing execution data, create internal findings, support action queues, prepare drafts for review and help the officer see where attention is needed.

This does not make the trade automatic.

It makes it traceable.

A trade is the bridge between the commercial promise and the operational responsibility.

Without a structured trade record, execution becomes reconstruction.

With a structured trade record, execution becomes workflow.

That is why Trades is not just another list in MN7R. It is the point where market work becomes accountable work.

Chapter 8. Clients

In commodity brokerage, a client is not just a name in a contact list.

A client can be a seller, a buyer, a processor, an exporter, an importer, a trading company, a logistics-linked counterparty, or a company that plays different roles at different times. The same company may appear in offers, bids, trades, confirmations, templates, reports, and execution follow-up.

If the client layer is weak, the whole workflow becomes weaker.

This is why MN7R treats Clients as an operating memory layer, not as a simple address book.

A client profile in MN7R can hold the company's working short name, formal names, client type, activity status, broker ownership, contacts, notes, business unit, and high-level legal, logistics, and banking sections. In a live production workspace, those details remain private and role-restricted. In this public guide, the important point is not the private content of the fields, but the function they serve.

They help the team preserve identity.

This matters because client names drift easily in real brokerage work. One broker uses a short name. Another uses a legal name. A third uses a translation. A fourth uses an old spreadsheet label. If those variants enter the market workflow as separate names, the system starts losing continuity.

MN7R is designed to reduce that drift.

When active client profiles are available, the system can use the current company identity in offer, bid, trade, template, and execution workflows. That means the seller or buyer selected in a market record is not just a piece of text. It is linked to a working company profile.

This affects several parts of the product.

In Deals, it helps brokers select the right seller or buyer when creating offers, bids, and trades.

In Clients, it helps maintain relationship context, client status, notes, labels, contacts, and ownership.

In EXE, it helps connect completed trades with structured contacts and document workflows.

In reports and analytics, it helps avoid duplicate or fragmented company views.

For brokers, client memory makes daily work faster and safer. They do not have to rebuild the same company context each time. They can work with a client as an ongoing relationship rather than a scattered sequence of messages.

For bosses, client memory supports supervision. A boss can see whether team clients are active, whether ownership is clear, whether follow-up is being maintained, and whether a relationship is drifting without attention.

For admins, client memory supports governance. Admin users can inspect broader client structure, help correct duplicates, review role visibility, and support onboarding.

A key design point in MN7R is broker ownership.

Commodity brokerage is relationship-driven. It matters who owns a relationship, who is assigned to a client, who can work with the client, and how visibility is handled across a team. MN7R's client model supports that by distinguishing ownership and visibility rather than treating every company as a flat public entry.

This is also why role scope matters. A broker may see and work the client records relevant to their operating scope. A boss may review all team clients. An admin may inspect broader support and governance context. This helps the system support teamwork without destroying relationship accountability.

The client profile also supports structured contacts.

In execution work, the right email contact or operational contact can matter. A business confirmation may need to go to a specific person or mailbox. A general company address may not be enough. A named contact may have a specific responsibility. A contact may be linked to business confirmation workflows.

When this information is structured, the system can support later workflows more reliably.

A client card can also carry legal, logistics, and banking sections. This does not mean public users or all roles see every sensitive detail. It means the product model recognizes that real brokerage work depends on more than a phone number and a name.

Legal identity matters.

Logistics context matters.

Banking context may matter.

Contacts matter.

Broker ownership matters.

The main product value is continuity.

A client should not be recreated again and again under slightly different names. A relationship should not disappear when a broker changes focus. An execution officer should not have to ask five people which company name is correct. A boss should not have to guess which broker owns the relationship.

MN7R turns the client book into a working part of the brokerage system.

It connects company identity to market flow.

It connects market flow to execution.

It connects execution back to relationship memory.

That is why Clients is not a side module. It is one of the structural supports of the whole workspace.

Chapter 9. EXE / Contract View

EXE is where completed trades become execution work.

This is one of the most important parts of MN7R because it deals with the long tail after the commercial agreement.

In many teams, the moment a trade is concluded feels like the end of the story. In reality, it is the beginning of another workflow.

The trade still needs to be confirmed. Contract or business confirmation documents may need to be prepared. Seller and buyer details must be correct. Delivery periods must be followed. Shipment and unload facts may appear over time. Invoices may need to be issued. Payment may be expected, delayed, or partially resolved. Commission may need review. Documents may need to be tracked. Someone must own the follow-up.

If this process is handled only through messages and separate files, the team easily loses visibility.

EXE / Contract View is MN7R's answer to that problem.

It acts as the execution register for completed trades. It lets users see which contracts exist, which are active, which are blocked, which need confirmation, which have invoice or payment pressure, which officer is assigned, and which records need attention.

The Contract View is not just a static list.

It is designed as a working surface. The list of contracts stays connected to the current action area. A user can select a contract and inspect the related execution workspace: overview, confirmations, drafts, delivery, invoicing, payment state, support notes, AI assistant context, and other follow-up surfaces depending on current rollout and role.

The public product idea is simple:

A completed trade should not become invisible.

It should remain part of a visible execution workflow until it is properly followed.

In EXE, a team can track several kinds of information.

There is confirmation status: has the business confirmation or required confirmation step been handled?

There is delivery balance: is the delivery period active, complete, in tolerance, final, or unresolved?

There is invoice status: has the invoice or related billing state been prepared, sent, pending, or unclear?

There is payment status and financial balance: is there an outstanding amount, a payment delay, or a state that needs review?

There is assigned officer context: who is responsible for following the execution work?

There are support notes: what practical information does the team need to remember about this contract?

There are filters and counters: which contracts are active, pending, blocked, or ready for follow-up?

For a broker, EXE keeps execution visible on deals where the broker participated. That means the broker does not lose sight of what happened after the trade.

For a boss, EXE is a supervision layer. It helps reveal whether the team's completed trades are moving through execution or accumulating risk.

For an admin, EXE is a support and review layer. It helps with onboarding, QA, role inspection, and operational control.

For an EXE officer, it is the main workspace.

The officer's job is not only to "watch contracts". It is to keep the execution process from dissolving into disconnected fragments. EXE helps by connecting contract records with confirmations, drafts, delivery, invoices, payment state, notes, actions, timeline, and AI support.

This is also where AI becomes most valuable.

The AI Officer Assistant is built around EXE because execution has many repeated patterns and many small gaps that can become large problems.

Missing buyer instructions.

Missing seller signature.

Missing buyer signature.

Shipped weight without unload weight.

Finalization without payment clarity.

Commission review candidate.

Payment delay.

Default risk signal.

These are not philosophical AI problems. They are everyday execution problems.

EXE gives the system the structure needed to detect them.

The result is not fully autonomous execution. That is not the goal. The result is better visibility, better follow-up, and less dependence on individual memory.

EXE turns post-trade work into a managed operating layer.

Chapter 10. Confirmations, Drafts and Templates

Execution work produces documents.

Business confirmations. Emails. Contract drafts. Addenda. Templates. Signed files. Follow-up messages. Internal summaries. Commission notes.

If these documents are created manually every time, errors appear quickly. A name is copied incorrectly. A subject line is inconsistent. A clause comes from an old version. A recipient is missing. A broker or officer edits the wrong draft. A team member cannot tell which version is current.

MN7R supports confirmations, drafts, and templates because execution work needs repeatable document structure.

A business confirmation is one of the practical bridges between a completed trade and the execution workflow. It helps put the deal terms into a structured communication for the seller, the buyer, or both sides.

In MN7R, confirmation work belongs to EXE follow-up, not to the market-posting phase. This distinction matters. A market post is about interest. A confirmation is about a concluded deal. The two should not be mixed.

The confirmation workflow can use structured context from the contract and client profiles. Recipient choices can be supported by client contacts where those contacts are available and marked for the relevant workflow. Subject and body content can be generated from templates and then reviewed by the user. If a confirmation was handled outside the system, users can still mark the status manually where the workflow allows it.

Templates reduce repeated manual writing.

A broker or officer should not have to rebuild common confirmation language from scratch every time. A reusable template gives the team a consistent starting point. It can be adapted to the counterparty, basis, workflow, and document type.

MN7R separates template maintenance from contract-level draft work.

This is important because templates are reusable assets, while contract drafts are specific to one contract or one execution situation. Mixing the two creates confusion. A template library should help maintain reusable language. A contract draft workspace should help create, review, and manage the actual draft for a particular deal.

The product also supports contract and addendum draft logic. A completed trade may need more than a confirmation email. It may need document drafts, signed file handling, addenda,

or generated contract-level assets. Keeping these connected to the selected contract helps reduce the risk that a document exists somewhere but is not visible in the execution workspace.

The practical workflow looks like this:

A trade is completed.

A contract record appears in EXE.

The user opens the contract.

The user prepares or reviews a confirmation, draft, or addendum.

Templates provide the starting structure.

Manual review remains visible.

The generated or attached document stays connected to the contract.

This is different from keeping templates in one folder and documents in another folder and messages in a third place.

MN7R's value is the connection.

The system is not trying to remove human review. It is trying to remove repeated copy-paste work and reduce preventable mistakes.

This matters especially when the same team handles many similar contracts. Similar does not mean identical. A DAP truck contract, an FCA rail contract, and an FOB sea contract can all involve confirmations and documents, but the details and risks differ. Templates give the team a base. The user still reviews the contract-specific result.

The AI draft layer builds on the same logic.

The AI Officer Assistant can prepare draft-only text for review: welcome messages, client emails, contract execution algorithms, commission messages, summaries, document requests, and needs-attention explanations. These drafts can have versions, validation warnings, feedback, and review states.

But the boundary remains explicit.

Draft generation is not sending.

A draft is not a legal commitment.

A generated text is not a final confirmation.

The human user remains responsible for review and use.

This is the right level of automation for execution support: reduce manual repetition, preserve context, and keep review visible.

Chapter 11. Secondary Views, Markets and Analytics

Commodity brokerage depends on context.

A broker rarely makes a decision from one record alone. The broker wants to know whether the market is active, whether a price is realistic, whether one basis is stronger than another, whether visible interest is concentrated in a specific corridor, whether historical prices support the discussion, and whether the team's own market flow tells a consistent story.

MN7R uses Secondary Views and analytics surfaces to support that context.

Secondary Views are not separate products. They are supporting views inside the current work area. Deals, Clients, and EXE can each use supporting views differently.

In Deals, secondary views can help brokers and supervisors understand the market before pricing, countering, posting, or explaining a physical idea to a client.

Market reference views can show global references, CBOT and MATIF context, and internal spot benchmark cards where available. These references help users see whether the broader market supports a current physical discussion.

Price views can show how selected commodity and basis combinations are moving over time. The user can compare curves, look at different time windows, and use historical context when enough data exists.

Index price views can support seasonal history and index-family context where data is available. Some prepared surfaces may be active now, while others become more useful as systematic index history accumulates.

Liquidity views help show where visible market interest is concentrated. This can help a broker or manager decide where there is enough flow to spend time and where a thin basis may not justify repeated effort.

Report views help turn filtered workspace data into repeatable market digests. Instead of manually rebuilding a report from scattered records, a user can work from saved profiles and structured filters.

Some surfaces may be active, prepared, hidden, or configured differently depending on the team and rollout stage. The important product idea is not that every chart is always visible. The important idea is that MN7R treats analytics as a support layer connected to the team's own operating data.

This is not a black-box price oracle.

MN7R does not claim that analytics alone decides the market. Physical commodity pricing depends on context: origin, basis, freight, quality, timing, liquidity, counterparty urgency, and broker judgement.

The purpose of analytics is to make the visible working data more useful.

A broker might ask:

Are we seeing more bids than offers in this basis?

Is this price in line with recent visible activity?

Is liquidity concentrated in one corridor?

Is the market moving or just one stale position?

Is the physical level rich or cheap against reference context?

Is there enough flow to keep pushing this route?

A boss might ask:

Which brokers are working the active corridors?

Which commodities have enough visible interest?

Where is the team spending too much time for too little flow?

Are matches turning into trades?

Are execution pressures rising in a specific route or commodity?

An admin or support user might ask:

Are reference labels consistent?

Are filters producing empty views because of missing data?

Are prepared surfaces working as intended?

Do role modes show the correct analytics context?

This is why analytics belongs inside the workspace, not outside it.

It supports the operating workflow.

When the team creates structured offers, bids, matches, trades, clients, and EXE records, those records can feed better visibility. The more disciplined the input, the more useful the analytics.

MN7R's analytics layer is therefore not separate from daily work. It is one of the reasons to structure daily work in the first place.

Chapter 12. Role Model: Broker, Boss, Admin and EXE Officer

MN7R is role-aware because brokerage work needs both speed and control.

A broker needs a fast daily workspace. A broker should be able to create and update offers and bids, review matches, create trades, work clients, follow execution on relevant deals, open manual help, use secondary views, and keep the market moving.

A boss needs broader team visibility. A boss is not just another broker. The boss needs to supervise flow, review team records, inspect client ownership, monitor execution pressure, coach brokers, correct mistakes, and see whether the team is moving useful opportunities forward.

An admin needs support and governance context. Admin users may need to inspect lower-role experiences, manage reference data, help onboard teams, review permission behavior, correct controlled dictionaries, support QA, and understand how different roles see the product.

An EXE officer needs execution focus. The officer works after the trade is completed. Their world is confirmations, signatures, instructions, delivery, documents, invoices, payment state, commission follow-up, support notes, drafts, action items, timeline memory, and exceptions.

These roles are connected, but they are not the same.

This is why MN7R separates assigned capability from selected role mode.

In public terms, this means a user may have a capability level, but the product can also let that user inspect or work in a selected mode where appropriate. For example, a boss may need to understand what a broker sees. An admin may need to inspect a lower-role experience for support or training. The system can support that without confusing the actual responsibility model.

The broker role is the base operating role.

It is closest to the market. Brokers create seller-side and buyer-side market records, work with clients, review possible matches, prepare trades, and follow execution where their deals are involved. The broker role must stay practical and fast.

The boss role adds team supervision.

A boss needs to see beyond one broker’s working scope. The boss cares about team-level market quality, flow, activity, client coverage, execution pressure, stale records, and operational follow-up. The boss does not necessarily need every admin control, but does need enough visibility to manage the desk.

The admin role adds governance and support.

Admin is about control, setup, QA, onboarding, role inspection, reference data, dictionary governance, and wider review. Admin users need access to areas that should not be part of a normal broker's daily workflow. For example, reference data governance belongs outside the broker operating role because changes to controlled dictionaries affect the whole workspace.

The EXE officer role is focused on execution continuity.

After a trade is completed, the EXE officer helps keep the contract process visible and disciplined. They need to see confirmations, delivery balance, invoices, payment status, financial balance, notes, assigned officer context, drafts, templates, AI suggestions, action items, and execution timeline.

Role separation protects the product from two opposite risks.

The first risk is under-control: everyone can change everything, and the workspace becomes unreliable.

The second risk is over-control: the system becomes so restricted that brokers cannot move fast.

MN7R tries to balance those risks.

Brokers get the working tools they need.

Boss users get team visibility and correction context.

Admins get governance and support tools.

EXE officers get execution focus.

AI features follow the same logic. The AI Officer Assistant is not exposed as a general chatbot for every user and every product area. It is focused primarily on EXE and protected role surfaces. Broker-only mode does not get the same assistant access. Public chat stays public-safe. Manual chat respects role boundaries.

This role-aware structure is not decoration. It is the product's safety model.

In commodity brokerage, the wrong user seeing the wrong control can create operational risk. The right user seeing the right context can prevent that risk.

That is why MN7R treats roles as part of the workflow, not just login labels.

Chapter 13. AI Officer Assistant

The AI Officer Assistant is the supervised AI layer of MN7R.

It is not designed as a general chatbot that floats over the whole product and says clever things. It is built primarily around EXE, because execution is where small missing details can create large operational problems.

A completed trade may look simple in the commercial register. Seller, buyer, commodity, basis, delivery period, price. But once it moves into execution, the work becomes more layered.

Has the seller signed?

Has the buyer signed?

Have buyer instructions been received?

Has shipment started?

Is shipped weight known?

Is unloaded weight known?

Is payment due?

Has payment been received?

Is there an outstanding amount?

Is the commission package ready for review?

Is the contract moving normally, or is it quietly becoming a risk?

These questions are repetitive, but they are not trivial. A human officer can check them manually, but when there are many contracts moving at once, small gaps are easy to miss.

The AI Officer Assistant exists to help with that exact problem.

It reads the structured execution context available in MN7R. It works with runtime facts, contract records, suggestions, action items, drafts, playbooks, approval states, timeline memory, and visible EXE scope. From that material, it helps the officer and supervisor understand what is missing, what is risky, what is waiting for review, and what should happen next.

The first layer of the assistant is deterministic.

This is important. Not everything in the AI layer depends on a language model. Many execution checks can and should be rule-based.

If seller signature is missing, the system can detect it.

If buyer instructions are missing while the contract is already in a shipment-sensitive stage, the system can flag it.

If shipped weight exists but unload weight is missing, the system can create a finding.

If payment is overdue or still unclear during finalization, the system can raise pressure.

If the contract appears physically advanced but commission has not been reviewed, the system can mark it as a commission-review candidate.

These are not “creative” AI tasks. They are structured execution checks. MN7R uses them to produce findings and suggestions that can become internal follow-up work.

The next layer is organization.

A finding by itself is not enough. If the system only says “missing buyer instructions”, the officer still needs to decide what to do with that fact. MN7R therefore connects findings to action items, SLA pressure, workbench states, drafts, playbooks, and timeline memory.

A finding can become a follow-up task.

A task can have priority.

A task can become overdue.

A contract can appear in a blocked, review, risk, or commission candidate state.

A draft can be prepared for human review.

A playbook can explain which stage the contract is currently in.

A timeline can show how the contract reached this point.

The assistant turns scattered execution signals into an internal operating structure.

This is why the assistant is called supervised. It does not silently take over the contract. It helps the human team focus.

The AI draft layer supports that same principle.

The assistant can prepare draft-only texts: welcome messages, client emails, document requests, contract summaries, commission messages, needs-attention explanations, broker clarification drafts, and execution algorithms. These drafts can be stored, regenerated, versioned, validated, reviewed, approved, rejected, or archived depending on workflow.

But a draft is not a send.

A draft is not a legal decision.

A draft is not a payment confirmation.

A draft is not a contract mutation.

The assistant prepares material for review. The user remains responsible for how that material is used.

This boundary is essential because commodity brokerage depends on trust and accountability. A system can help write a clean message, but it should not silently decide to send a client-facing communication. It can detect payment pressure, but it should not confirm a payment. It can show that a contract may be ready for commission review, but it should not issue financial instructions without human control.

The AI Officer Assistant also includes a chat interface for allowed internal EXE users. The chat can answer questions about the selected contract, the EXE workflow, findings, action items, drafts, playbooks, timeline memory, and the assistant's own boundaries.

A user can ask:

"What should I do with this contract?"

"Why is buyer instructions missing?"

"How do I prepare a commission message?"

"Why is this draft blocked by validation?"

"What does this playbook stage mean?"

"Why is this contract ranked as high priority?"

The assistant should answer from the current EXE context rather than from a generic manual. If a contract reference is mentioned, such as a visible contract number, it should try to resolve that reference inside the permitted scope. If the user asks for an action that is forbidden, such as payment confirmation, the assistant should refuse and offer a safe alternative, such as creating an internal review task.

The assistant also supports product improvement through Codex task candidates. If a user reports that a needed function is missing, or that a workflow is broken, the assistant can help structure the problem into a product gap. That does not mean the assistant changes the code. It means the operational pain can be captured in a useful format for later development review.

This is one of the deeper ideas behind the MN7R AI layer.

The assistant is not only there to answer questions. It is there to help the product learn where the workflow breaks.

If officers repeatedly miss the same evidence, the system can show coverage gaps.

If recommendations are repeatedly marked wrong priority or missing context, the policy loop can learn that the ranking needs adjustment.

If a playbook stage often lacks the same fact, the team can improve either process discipline or product fields.

If a runtime surface degrades, admin diagnostics can show which AI capability is healthy, degraded, unavailable, or intentionally fallback-only.

This makes the AI Officer Assistant more than a drafting tool. It is part of the execution operating system.

It sees execution signals.

It organizes follow-up.

It prepares reviewable drafts.

It explains current context.

It preserves memory.

It helps identify product gaps.

It remains inside human review boundaries.

That is the intended balance.

MN7R does not present AI as a magic replacement for execution people. It presents AI as a supervised execution layer that helps one good officer or supervisor handle more contracts with better focus, fewer missed follow-ups, and clearer operational memory.

Chapter 14. Action Queue, Workbench, Playbooks and Timeline

The AI Officer Assistant becomes useful when its signals turn into work.

A warning is not enough. A dashboard is not enough. A list of findings is not enough.

Execution work needs a queue.

MN7R therefore connects the assistant to an internal action queue and SLA layer. The purpose is to make the next operational step visible.

If buyer instructions are missing, the system can create or suggest an action item.

If seller signature is missing, the action queue can show that follow-up is needed.

If unloaded weight is missing after shipped weight exists, the system can show that the contract cannot fully advance.

If payment is overdue, the item can receive higher pressure.

If a commission message may be ready for review, the system can surface it without declaring that commission must be issued.

The action queue is internal. It is not a client communication channel. It does not mutate contract terms. It does not confirm payment. It does not mark a contract executed. It organizes what the officer or supervisor needs to look at.

This is where SLA becomes practical.

In commodity execution, not every missing item has the same urgency. A missing playbook fact before shipment is not the same as an optional note. A missing buyer instruction during active shipment is not the same as a low-priority documentation improvement. A payment delay is not the same as a draft waiting for review.

SLA state helps separate:

what is on track;

what is due soon;

what is overdue;

what is missing proof;

what is blocked;

what needs human review.

The workbench sits above this queue.

If the action queue shows individual actions, the workbench shows the operating state of contracts.

It can group contracts into buckets such as blocked, ready for review, needs correction, drafts waiting review, potential risk, commission review candidate, missing playbook, or watching. This lets a supervisor see not only the list of tasks, but the shape of the execution desk.

The difference matters.

An officer may have twenty open actions. The supervisor does not always need to read all twenty. The supervisor needs to know which contracts are blocked, which are waiting for draft review, which have payment pressure, which are missing required evidence, and which have moved forward automatically because a previously missing fact appeared.

The workbench gives that higher-level view.

Playbooks add process structure.

A playbook is the expected route for a certain kind of execution. In MN7R, playbooks can be linked to basis, transport, and contract stage. A DAP truck contract does not behave like an FOB sea contract. FCA rail has different operational expectations from EXW pickup. CIF sea carries different context from CPT.

A playbook can define expected stages, required facts, evidence hints, default internal actions, red flags, and safe next steps.

For example, a DAP truck flow may require contract review, signatures, buyer delivery instructions, shipment schedule, shipped weight, unloaded weight, payment state, and commission follow-up.

If buyer instructions are missing, the assistant does not see that as an isolated empty field. It understands that for that playbook and stage, the missing fact may block progress.

This is the step from field-checking to process intelligence.

The assistant is no longer asking only:

“Is this field empty?”

It can begin to ask:

“Can this contract advance through its expected execution path?”

Playbooks also make drafts more useful. A generic email draft is less valuable than a draft prepared in the context of a specific playbook stage. A commission message is more useful if it understands whether delivery and payment facts support review. A needs-attention explanation is stronger if it can say that a required fact is missing for the current stage.

Timeline and execution memory complete the picture.

A contract is not only its current state. It has a history.

A runtime fact was updated.

A suggestion was created.

An action item was opened.

A draft was generated.

A playbook was assigned.

An approval request was reviewed.

A note was added.

A finding was resolved.

A contract appeared in a daily digest.

The timeline gives the officer and supervisor a readable execution memory. Instead of reconstructing the past from messages, the user can see how the contract moved.

This becomes especially important as the assistant gains more responsibility for internal operating work. If the AI layer creates an action item, auto-resolves it when evidence appears, prepares a draft, or ranks the contract as high priority, the team needs to understand why.

Timeline is the audit memory for that work.

A future supervisor should be able to ask:

“Why was this contract blocked?”

“When did the payment pressure appear?”

“Who reviewed the draft?”

“What did the AI resolve automatically?”

“Which playbook stage was incomplete?”

“Which evidence was missing?”

“What changed since yesterday?”

The answer should not require detective work across five systems.

It should be visible in the contract memory.

Action Queue, Workbench, Playbooks and Timeline therefore form one operating loop.

The playbook says what should happen.

The runtime facts show what has happened.

The assistant detects gaps.

The action queue turns gaps into work.

The workbench shows contract-level pressure.

Drafts help prepare reviewable text.

The timeline preserves what changed.

The supervisor sees where to focus.

This is how MN7R moves from a monitor to an execution operating system.

Chapter 15. Manual AI Chat, Public AI Chat and Demo Assistant

MN7R has several AI chat surfaces, but they are not the same tool with the same permissions.

This matters because the wrong kind of AI chat can create risk. A public visitor should not see private workspace context. A broker should not receive admin-only instructions. A general product question should not be answered as if the user were inside a protected contract. An EXE officer asking about a selected contract needs a different kind of answer from a website visitor asking what MN7R is.

MN7R therefore separates AI chat surfaces by context and role.

The EXE Assistant Chat is the protected operational chat for allowed internal EXE users. It lives in the execution context. It can answer questions about selected contracts, runtime facts, suggestions, action items, drafts, playbooks, timelines and boundaries. It can resolve visible contract references within the user's permitted scope. It can explain why something is missing, why a draft has a warning, why a contract is blocked, or why an action is ranked as urgent.

It can also propose confirmable internal tool actions where allowed.

But it does not become an unrestricted actor. It does not send client messages automatically. It does not confirm payments. It does not change contract terms. It does not cross role boundaries.

Manual AI Chat is different.

It lives inside the manual experience. Its purpose is to help users understand the product without leaving the documentation. It must respect role boundaries. A broker should not receive hidden admin-only operational instructions. A boss should not be shown controls that only admin users can manage. An admin can inspect a broader capability set, but still within documented product scope.

Manual AI Chat is therefore a role-scoped knowledge helper.

It answers:

"Where is this function?"

"What can my role do here?"

"Why can't I see this control?"

"What happens after this action?"

"Which manual page explains the workflow?"

This reduces training friction. Instead of searching the entire manual, the user can ask a contextual question.

Public Landing AI Chat is narrower again.

It is public-safe. It is available without workspace authentication. It answers questions about the product using public documentation and public product wording. It can explain modules, roles, EXE, AI boundaries, public demo, books, guides, and the current documented scope. It does not expose protected workspace internals. It does not access private records. It does not know real clients, real trades or private execution data.

This allows a visitor to ask:

“What is MN7R?”

“Who is it for?”

“What does the AI Officer Assistant do?”

“Does the public demo use real data?”

“What does MN7R not automate?”

The answer should be useful, but safe.

The Demo Assistant is also public-safe, but in a different way.

It works with synthetic demo data. It helps visitors understand the demo workspace: offers, bids, matches, trades, EXE records, AI suggestions, action items, drafts, playbooks and guided scenarios. It can simulate the experience of asking what to do with a synthetic contract. It may show how a health check would work or how a draft would be prepared in demo mode.

But it is not connected to real production records.

It does not call real protected APIs.

It does not create real Codex tasks.

It does not send Telegram or email.

It does not expose private workspace context.

The common rule across all chat surfaces is simple:

the assistant must match the user’s context.

A public visitor gets public-safe product explanation.

A manual user gets role-safe product help.

An EXE user gets protected execution context.

A demo visitor gets synthetic demo context.

This may seem conservative, but it is the correct design for a product that handles commercial, client, execution and payment-related workflows.

AI assistance is valuable only if it respects authority.

A good assistant does not merely answer. It knows what it is allowed to know, what it is allowed to suggest, what requires confirmation, and what is forbidden.

Chapter 16. Public Demo and Synthetic Data

MN7R includes a public demo because a product like this is difficult to understand from screenshots alone.

A buyer of software can read a feature list, but brokerage workflow is practical. A broker wants to see offers, bids, matches and trades. A supervisor wants to see team visibility. An execution user wants to see contract follow-up. A potential partner wants to understand how AI support appears inside the workflow without touching real data.

The public demo exists to make that possible.

It is built on synthetic data only.

That means the demo does not use real clients, real brokers, real trades, real EXE records, real prices from production, real payments, real banking data, real Telegram flows, or private workspace events.

The purpose is to show product behavior safely.

The demo can display a compact workspace with Deals, Clients, EXE, AI Assistant, analytics and guided scenarios. Visitors can switch between demo perspectives such as Demo Broker, Demo Boss, Demo Admin and Demo EXE Officer. These role modes simulate product perspectives. They are not real accounts and do not grant access to protected data.

The demo can show synthetic offers and bids.

It can show a synthetic match.

It can show a synthetic trade.

It can show a synthetic EXE contract.

It can show an AI suggestion such as missing buyer instructions.

It can show a simulated action item.

It can show a simulated draft.

It can show a synthetic playbook or timeline event.

The key word is simulated.

Demo actions are local and safe. If the visitor clicks “Run health check”, the demo can calculate suggestions over synthetic data. If the visitor clicks “Generate draft”, the demo can produce deterministic demo text. If the visitor creates an action item, it is a browser-local simulation. If the visitor explores a guided scenario, the system uses artificial scenario anchors.

Nothing is written to production.

No protected API is called.

No real email is sent.

No real Telegram message is sent.

No real Codex candidate is created.

No real LLM provider is required.

This separation is important because MN7R needs both transparency and confidentiality.

Potential users should be able to understand the system. They should see that the product is real, structured and operational. But real brokerage data is sensitive. Real client names, trade routes, payment states, and execution problems must remain private.

Synthetic demo data solves that tension.

It lets the product show the shape of work without exposing the work itself.

The demo is also useful for onboarding. A new user can see how the workflow is supposed to feel before entering a protected workspace. A manager can understand how roles differ. A prospect can see how Deals connect to EXE. A non-technical stakeholder can understand why AI support is supervised rather than autonomous.

Guided scenarios make this easier.

A “Find a match” scenario can show how an offer and bid may overlap.

A “Create trade” scenario can show how a matched idea becomes a trade record.

A “Track EXE” scenario can show how a completed trade becomes contract follow-up.

An “AI detects missing buyer instructions” scenario can show why structured execution facts matter.

The public demo is therefore not a toy version of MN7R. It is a safe explanation layer.

It supports evaluation without access.

It supports learning without risk.

It supports product storytelling without exposing private data.

Together with the public guide, blog, manual excerpts, and books, it helps external readers understand the same principle:

MN7R is not only a monitor.

It is a structured operating workspace for commodity brokerage teams.

Chapter 17. What MN7R Does Not Do Automatically

A good operating system must be clear not only about what it does, but also about what it refuses to do.

This is especially important in commodity brokerage.

The work involves commercial relationships, legal documents, payment timing, execution obligations, banking details, contractual terms, and reputation. A system that helps the team move faster is valuable. A system that silently makes sensitive decisions without the right human control is dangerous.

MN7R is designed around supervised automation.

The product can help detect problems, organize work, prepare drafts, show evidence gaps, suggest next actions, build contract memory, and surface execution pressure. But several categories of action remain protected by design.

MN7R does not automatically send client communications.

The system can help prepare a draft. It can help structure a business confirmation. It can help produce a document request, a contract summary, a commission review message, or a needs-attention explanation. It can show that a message may be useful.

But preparing text is not the same as sending it.

Client communication carries commercial and legal meaning. A message can affect trust, negotiation, timing, and liability. For that reason, external communication must stay under human control. MN7R can support the workflow, but it does not silently speak to clients on behalf of the team.

MN7R does not automatically confirm payments.

Payment status is sensitive. A payment may be expected, pending, partial, delayed, confirmed elsewhere, or still under review. The system can show payment-related fields, payment pressure, financial balance, overdue signals, or a need for review. It can help the officer see that payment needs attention.

But the system does not decide that payment is confirmed without the appropriate human or controlled operational process.

MN7R does not automatically change bank details.

Banking details are among the highest-risk data categories in any commercial workflow. A wrong account, stale beneficiary detail, or unauthorized change can create serious financial risk. MN7R may help organize client profiles and related information, but any change to sensitive banking data belongs under strict governance and human-controlled review.

MN7R does not automatically change contract terms.

Contract terms are not ordinary metadata. Price, volume, basis, delivery period, payment terms, parties, requisites, and special conditions have commercial and legal consequences. MN7R can help expose inconsistencies, prepare drafts, record structured data, and show review states. But it does not silently rewrite the commercial agreement.

MN7R does not automatically mark contracts executed.

The final state of a contract must reflect real completion, not only system confidence. A contract may appear nearly complete because some data is present, but there may still be unresolved documents, payments, quality issues, commission questions, or manual confirmations. The system can show that a contract may be close to completion. It can help identify missing steps. It can reduce the noise around review.

But the decision that a contract is fully executed remains a controlled human or governed workflow decision.

MN7R does not automatically trigger default or termination decisions.

Potential default risk can be detected. Missing evidence can be highlighted. Payment delays can be surfaced. Serious overdue action items can be escalated inside the operating view. But default and termination are not AI decisions. They require human judgement, legal context, commercial understanding, and proper documentation.

MN7R does not expose private production data in the public demo.

The public demo is built with synthetic data. It exists so visitors can understand how the product works without accessing real clients, real trades, real prices, real routes, real payments, real banking details, or private workspace records.

The same principle applies to public product material. Public guides, website pages, and public chat surfaces explain the product, but they do not reveal private workspace data.

These boundaries are not a weakness of the system. They are part of the product design.

The goal is not to make the AI layer “do everything”. The goal is to let it take over low-risk internal operating work while keeping sensitive decisions in the right hands.

It can create internal structure.

It can help officers see missing information.

It can prepare drafts.

It can help prioritize contracts.

It can preserve execution memory.

It can make the team more disciplined.

But it does not replace commercial responsibility.

That is the right balance for a brokerage workspace.

Chapter 18. Why This Matters for Broker Teams

Brokerage teams do not usually fail because they lack effort.

They fail because the work becomes too fragmented.

A broker remembers a seller. Another broker remembers a buyer. A supervisor remembers a previous deal. An execution officer remembers a payment issue. A client name is stored in one format in one place and another format somewhere else. A price was discussed in a message. A delivery instruction was sent in a different chat. A commission note is in a spreadsheet. A trade is “done”, but no one is quite sure which execution step is next.

The team is working hard, but the system is not holding the work together.

MN7R matters because it turns fragmented brokerage activity into a connected operating flow.

The first benefit is fewer missed follow-ups.

When offers, bids, matches, trades, clients, and execution records are structured, the team has more chances to notice what needs attention. A stale record can be seen. A match can be reviewed. A completed trade can enter EXE. A missing instruction can become an action. A payment delay can become visible before it becomes a crisis.

The second benefit is clearer ownership.

Brokerage depends on relationships. It matters who owns the client, who worked the seller side, who worked the buyer side, who is responsible for execution, and who needs to review a task. MN7R makes these roles more visible. It does not remove human relationships from brokerage. It helps the team manage them with less ambiguity.

The third benefit is stronger market-to-execution continuity.

In many teams, the commercial workflow and the execution workflow live in different worlds. A broker closes a trade, and then an execution person starts a separate process. This creates a break in context. MN7R reduces that break by letting completed trades feed EXE and by keeping contract follow-up connected to the original commercial record.

The fourth benefit is better supervision.

A boss cannot manage a team only by asking for updates. They need visibility into market flow, stale records, match quality, client ownership, execution pressure, payment risk, and team follow-up. MN7R gives supervisors a way to see the work as a system, not only as individual reports from brokers.

The fifth benefit is faster onboarding.

Commodity brokerage has many terms, many exceptions, many habits, and many hidden rules. A new broker or officer needs to understand not only what to click, but how the work fits together. A structured workspace, role manuals, product guides, public demo, and AI-assisted explanations make onboarding less dependent on oral tradition.

The sixth benefit is operational memory.

A brokerage business loses value when knowledge stays inside individual heads. MN7R helps preserve more of that knowledge inside structured records: clients, offers, bids, trades, EXE facts, notes, drafts, timeline events, playbooks, and action items. This does not eliminate the need for experienced people. It gives experienced people a better operating surface.

The seventh benefit is supervised AI support.

AI is useful in brokerage only if it respects the workflow. A generic chatbot can produce fluent text, but fluency is not the same as operational reliability. MN7R's AI layer is grounded in the execution workspace. It reads structured facts. It creates findings. It organizes action items. It helps prepare drafts. It explains the current context. It supports timeline memory. It respects role boundaries and forbidden actions.

This is different from asking a general AI model to "help with a contract".

The assistant works inside the product's operating structure.

That makes it more useful and safer.

For broker teams, the strategic value is simple: MN7R helps the team keep speed without losing control.

Brokers still work quickly.

Managers get visibility.

Admins keep governance.

Execution officers get structure.

AI supports routine operating focus.

The team spends less time reconstructing what happened and more time deciding what should happen next.

That is the real value of a brokerage workspace.

Chapter 19. Current Capability Snapshot

The following snapshot summarizes the public-safe product areas of MN7R and how they support a brokerage team.

Deals

Deals is the main market workspace. It is where brokers create and work with offers, bids, matches, and trades. It is the daily operating surface for market flow. For a broker, this is where ideas become structured records. For a boss, it is where team activity becomes visible. For an admin, it is where support and role review can be performed.

Why it matters: Deals turns live market work into reusable operating data.

Offers

Offers capture seller-side market interest. A seller-side idea can include commodity, origin, crop year, quantity, tolerance, basis, delivery place, transport, delivery period, price when available, currency, payment terms where relevant, VAT context, notes, broker identity, and publication preview.

Why it matters: offers stop being temporary messages and become records that can be updated, matched, reported, converted, and later followed.

Bids

Bids capture buyer-side market interest. A bid can include buyer, commodity, quantity or open quantity, destination, basis, transport, delivery period, price when available, and other commercial context.

Why it matters: buyer demand remains visible for matching, supervision, trade creation, and later execution continuity.

Matches

Matches compare buyer and seller interest. They help reveal where a commercial opportunity may exist. They do not replace judgement and do not automatically create a deal.

Why it matters: matches reduce the chance that a workable seller-buyer pair is missed because information is scattered.

Trades

Trades record the commercial closure point. A trade connects seller, buyer, seller-side broker, buyer-side broker, commodity, basis, route, volume, price, period, and commission context where known.

Why it matters: trades bridge the market workflow into execution.

Clients

Clients is the relationship and company memory layer. It helps manage company names, client status, broker ownership, contacts, legal/logistics/banking context at a role-safe level, and linked history.

Why it matters: client memory reduces duplicate identity drift and keeps relationships connected to market and execution work.

EXE / Contract View

EXE tracks execution after a trade is completed. It supports contract records, confirmations, delivery balance, invoice status, payment status, financial balance, assigned officers, support notes, drafts, filters, and execution pressure.

Why it matters: EXE makes post-trade work visible instead of letting it dissolve into messages and files.

Confirmations, Drafts and Templates

This layer supports business confirmations, reusable templates, contract drafts, addenda, generated documents, and signed-file handling where enabled.

Why it matters: execution teams can reduce repetitive writing and document errors while keeping human review visible.

Secondary Views and Analytics

Secondary Views provide supporting market and operational context: market references, price views, index prices, liquidity, reports, and other analytics surfaces depending on rollout and role.

Why it matters: analytics helps brokers and managers understand pricing context, visible flow, basis concentration, and team focus.

Role Model

MN7R supports Broker, Boss, Admin, and EXE Officer perspectives. Each role has a different operating purpose and visibility model.

Why it matters: brokerage teams need speed and control at the same time.

AI Officer Assistant

The AI Officer Assistant is a supervised EXE-first AI layer. It supports deterministic checks, findings, action queue, workbench, drafts, playbooks, next best action, progress scoring, approvals, timeline memory, runtime diagnostics, and EXE chat for allowed users.

Why it matters: the AI layer helps organize execution without taking over sensitive decisions.

Manual AI Chat

Manual AI Chat helps users ask role-aware questions inside the documentation experience.

Why it matters: users can understand product functions without leaving the help context, while role boundaries remain protected.

Public Landing AI Chat

Public AI Chat answers public-safe questions about MN7R, modules, roles, AI boundaries, demo, and documented product scope.

Why it matters: visitors can learn about the product without accessing private workspace data.

Public Demo

The public demo shows synthetic Deals, Clients, EXE, AI Assistant and guided scenarios.

Why it matters: prospects can explore product behavior safely without seeing production data.

Public Guides and Books

MN7R publishes product and educational guides for public readers. These materials explain the product, the market context, and the brokerage workflow without exposing private workspace data.

Why it matters: teams can learn the product and the market before onboarding into the protected workspace.

The snapshot is not a replacement for the live product. It is a public map of what MN7R currently supports and why those capabilities matter.

Chapter 20. How to Start with MN7R

The easiest way to understand MN7R is to approach it in layers.

First, understand the workflow.

The core workflow is Offers → Bids → Matches → Trades → EXE. If that chain is clear, the rest of the product becomes easier to understand.

An offer is seller-side interest.

A bid is buyer-side interest.

A match is a comparison.

A trade is commercial closure.

EXE is execution follow-up.

Second, explore the public demo.

The demo is designed for safe evaluation. It uses synthetic data and does not require access to a protected workspace. It can show how the main areas of the product fit together: Deals, Clients, EXE, AI Assistant, analytics, and guided scenarios.

A good first demo path is:

look at offers and bids;

review a match;

see how a trade connects to EXE;

open a synthetic EXE contract;

inspect AI suggestions;

look at a draft or playbook;

follow a guided scenario.

This will not show production data, but it will show the shape of the workflow.

Third, read the brokerage guides.

MN7R is not only a software interface. It belongs to a market with its own language: basis, freight, logistics, offers, bids, execution, documents, payment timing, and broker responsibility. The public brokerage guides help new users understand this market context.

Fourth, read this Product Guide as the product map.

This guide explains why MN7R exists, how the major product areas connect, what the AI layer does, and where the system keeps human review in control.

Fifth, define your team structure.

Before onboarding into a protected workspace, it helps to know who will play which role:

Who are the brokers?

Who supervises the team?

Who manages reference data and support?

Who follows execution?

Who reviews drafts?

Who handles payments and commission review?

Which workflows are land brokerage, sea brokerage, freight, trade solutions, or mixed?

The product works best when the role model matches the real team.

Sixth, start with a narrow operating flow.

A team does not need to configure everything on day one. A practical start might focus on:

offers and bids;

client identity;

matches and trades;

EXE follow-up;

AI suggestions;

action queue;

basic reports.

Then the team can expand into deeper templates, analytics, playbooks, automation policies, and advanced review flows.

Seventh, keep the system disciplined.

MN7R becomes more valuable when records are maintained. If offers and bids are stale, matching becomes weaker. If clients are duplicated, relationship memory suffers. If trades are incomplete, EXE starts with missing context. If runtime facts are not updated, the AI layer cannot help accurately.

The product supports discipline, but the team still needs to use it as an operating system.

Eighth, keep human control where it belongs.

MN7R can help make work visible. It can prepare drafts. It can rank attention. It can detect missing facts. It can preserve memory. It can support onboarding and supervision.

But the brokerage team remains responsible for commercial judgement, client communication, legal commitments, payment confirmation, bank details, contract terms, and final execution decisions.

That is the correct relationship between software, AI, and brokerage responsibility.

To start with MN7R, use the public materials first:

read the guide;

explore the demo;

review the brokerage education materials;

contact the MN7R team;

discuss your workflow;

then configure the protected workspace around your real team, roles, clients, commodities, routes and execution needs.

MN7R is not meant to replace a brokerage business.

It is meant to give that business a stronger operating structure.

Afterword. A Living Guide

This Product Guide is a living document.

MN7R is changing continuously. New product surfaces appear. Existing workflows become sharper. AI assistant capabilities become more disciplined. Demo scenarios improve. Documentation is updated. Some features become more public-facing. Others remain internal or role-protected. Some prepared capabilities may become active for a team only after configuration or rollout.

For that reason, this guide should not be read as a final frozen book.

It is a public snapshot of MN7R at the time of generation.

The next version may describe new functionality, improved workflows, clearer role boundaries, additional demo scenarios, better AI operating support, new analytics, or updated safety rules. Older descriptions may be removed or adjusted when the product changes.

The goal is to keep the guide useful, factual, public-safe and current.

If you are reading this as a broker, we hope it helps you see how your daily market work can become more structured without losing speed.

If you are reading this as a team lead, we hope it helps you imagine better visibility across brokers, clients, execution and follow-up.

If you are reading this as an execution officer, we hope it shows how structured records and supervised AI can reduce the burden of chasing scattered information.

If you are reading this as a potential partner or client, we hope it gives you a clear picture of what MN7R already does and why the product exists.

Thank you for reading.

We built MN7R for teams that know commodity brokerage is not just about finding a price. It is about carrying a deal from first market signal to completed execution, without losing context along the way.

That is the work MN7R is designed to support.