BROKER CONSOLE · MODULE REFERENCE

Server Management

The machines underneath: restarts on a clock rather than on somebody remembering, the logs and the watchdogs that say something is wrong, and every service this installation talks to.

Agent on the broker's own machine · Dry run first, always · Market-hours guard · Silence is never green · Prometheus and Zabbix export · No geolocation database

PRODUCT	MODULE	CONTENTS	VERSION
HonorPro Hub	Server Management	3 menus · 18 tabs	1.0 · 24 Sep 2026

Scheduling	MT5 Scheduled Restarts · MT4 Scheduled Restarts
Alert Systems	Notification Channels · Error & Config Logs (MT5) · Error & Config Logs (MT4) · DataCenter Notifier (MT4) · MT5 IP Geolocator · MT4 IP Geolocator · Watch Dog · Metrics Export
Platform Services	Accounts · Orders · History · Groups · Audit log · Operators · Connections · Agent

01 THE MODULE

One daemon on the broker's machine, and nothing else

This console runs in a data centre the brokerage does not own. Everything it knows about the trading host, and everything it can ask that host to do, travels down one outbound connection from a small daemon the broker installed themselves. The Hub plans and records; the Agent executes. No screen in this module runs a command.

3
MENUS
Scheduling, Alert Systems, Platform Services

18
TABS
Each one an address a person can paste to a colleague

6
AGENT CAPABILITIES
Declared by the Agent, never assumed by the Hub

The Agent's own config is the boundary

The services this console can name are the ones listed under **services** in the Agent's configuration file on the broker's machine. There is no free-text service box anywhere in this module. A key that is not in that list is refused by the Agent with its reason, and refused by the Hub before a job goes down the wire at all. Only somebody with a shell on that box can widen it.

- **services.inventory** — describe the services on this machine
- **service.restart** — stop and start one of them, in order
- **logs.tail** and **logs.sample** — the server's own log files
- **logins.read** — the login lines out of those same files
- **dc.status** — an MT4 data-centre ring

Every command the Agent runs is a fixed argument list, never a shell string. Nothing out of a job is interpolated into a command line, so **a service name is a service name and not an injection point**.

Silence is never green

A service this console has not heard from is never drawn as healthy. It gets its own count, the fault colour rather than a blank, and its own sentence — "the last thing known about this service is 40 minutes old". Silence and health look identical on a badly built dashboard, and that is exactly how a green board ends up sitting above an outage.

The same rule off-console

The metrics export writes **service_up** only for a service whose state is current. Anything else gets **service_stale** and its age, and no **_up** sample at all — because a gauge that keeps its last value after the thing producing it went quiet is how a NOC is misled by its own monitoring.

Never is not zero

The Agent reports every thirty seconds, so an absence is measured and named: "silent for 14 minutes". An Agent that enrolled and has never reported says exactly that, because it is a different fault from one that reported yesterday and stopped.

02 MENU · SCHEDULING

A restart made hard to do by accident

A restart disconnects every client on the box. Both tabs are the same screen and the same mechanism with the platform pinned in the address, and every guard below is enforced on the server — a console that were the only thing stopping a bad restart would stop it right up until somebody used curl.

MT5 Scheduled Restarts `/server/restarts/mt5`

The schedules this installation keeps, each a named window and an ordered list of services with a wait between them. "Plan it" is the primary button and the only one-click action: it asks the broker's machine what would happen, step by step, and shows it. A step naming a service the Agent has not reported is planned as *nothing*, with that reason, rather than being sent to see what happens.

Dry run by default · Ordered steps with waits · Up to 12 hours · Run history

MT4 Scheduled Restarts `/server/restarts/mt4`

The same screen and the same mechanism for the other platform — a service is a service, and the Agent restarts an MT4 unit exactly as it restarts an MT5 one. This installation has no MT4 host to point it at, and the screen says so on its face rather than offering a schedule that could never run.

Same mechanism · No MT4 host declared

The confirmation phrase

A live run is refused unless the operator types back the exact sentence the plan printed — the services and the window, spelt out. It is not a checkbox, because a checkbox is something a hand does without a mind attached, and it is not a phrase this console could compose on its own, because a confirmation the caller can generate is not a confirmation.

Market hours

Every minute of the window is checked against the brokerage's own book, not just its two ends — a window that starts after the Friday close and runs past the Sunday open is a real thing. A clash is refused in a sentence naming the minutes that are inside and what it means for clients. The default book is Sunday 22:00 to Friday 22:00 UTC and the brokerage sets its own.

Overriding the market-hours refusal takes a reason, and the reason is written onto the run record beside the operator's name — it is the first thing anybody asks for afterwards. A schedule granted standing permission to run inside market hours has to carry the same reason before it can be saved.

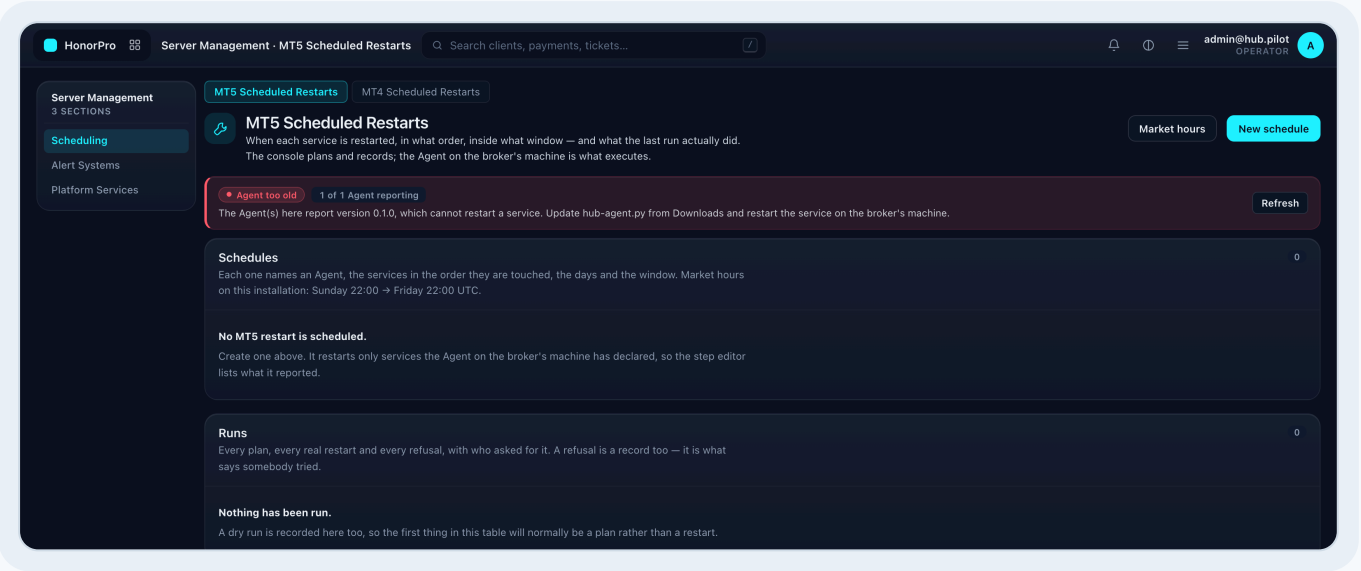
A restart — a dry run included — is refused when no Agent is enrolled, when the Agent has never reported, when it last reported more than two minutes ago against a thirty-second heartbeat, or when it is too old to carry the ops channel. Each refusal names what to do.

A run ends as **planned**, **running**, **succeeded**, **partial**, **failed** or **refused**. A refusal is kept as a run of its own: it is the record that says somebody tried.

03 PLANNING A RESTART

The screen says what it cannot do first

MT5 Scheduled Restarts on a live installation. The band under the title is the first thing on the page and it is about the Agent rather than the schedule: what is on the broker's machine decides what this screen may offer.



Scheduling · MT5 Scheduled Restarts. The Agent on this pilot is an older build without the ops channel, so the screen says so in that version's own terms and names the fix, rather than failing when somebody presses Plan.

The market-hours line

"Market hours on this installation: Sunday 22:00 → Friday 22:00 UTC" is the book every window is checked against, minute by minute. It is printed where a schedule is written rather than buried in a settings screen, and a brokerage with a different book sets it from the same page.

Empty is explained

Neither table is left blank. A schedule restarts only the services the Agent has declared, so the step editor lists what it reported; and the run table normally opens with a plan rather than a restart, because a dry run is recorded here too.

04 MENU · ALERT SYSTEMS

What tells a person the platform is in trouble

Eight tabs, one delivery component and one rule: an alert fires on a change of state, not on every sweep, because a desk told each minute that the same service is still down stops reading the channel.

Notification Channels `/notifications`

Where this brokerage is told things. One entry in the menu rather than one per channel, because binding the ops chat is a single decision. A channel is bound by having staff message the brokerage's own bot with a code this screen issues; the code lives fifteen minutes, and only a chat that completed the handshake is ever written to.

Alerts are recorded as events whether or not anything is bound, and each is closed by its own recovery — so the desk sees "down" and "back" as a pair. Open alerts and bound channels are both published to the metrics export, which is how a brokerage finds it has alerted nobody.

Bind by code · 15-minute code · Raised and resolved · Shared by every feature here

Error & Config Logs (MT5) `/server/logs/mt5`

The division of labour is the point of this screen and it is stated on it: the Agent tails the server's log directory because that is the only place those files exist, and the patterns that decide what a line *means* live here, where an operator can read them, edit them and test them. A pattern in a config file on a Windows box in another country is a pattern nobody ever changes.

Five starter patterns ship per platform — service crash, database or disk failure, repeated login rejection, configuration rejected, feed or bridge disconnected — each with a severity, an hourly ceiling so one service in a restart loop cannot empty the desk's attention, and a count of what went over that ceiling rather than a silent drop. A pattern is tested against the lines the Agent actually delivered before it can be saved, and the test says plainly when it matched every line in the sample, which is the commonest way a log monitor is broken.

5 starter patterns · critical · warning · info · Hourly ceiling · Tested before saved · Suppressed counted

Error & Config Logs (MT4) `/server/logs/mt4`

A log file is a log file: the same screen, the same patterns and the same tail, pointed at an MT4 directory. The starter patterns are written for both servers' vocabulary. An empty match table is always drawn with the reason it is empty — "no errors" and "nobody is reading the log" look the same and mean opposite things.

Same screen · Why it is empty, always

DataCenter Notifier (MT4) `/server/datacenter`

An MT4 installation is a main server with a ring of data centres in front of it. A centre that *drops* sheds its clients quietly; a centre that *lags* leaves them trading on stale prices. So the three states are connected, lagging and dropped, and what the screen shows is the transitions — with the time, and with whether anybody was told. This installation has no MT4 server, which is said in the band at the top in as many words, and the screen names the one line of Agent configuration that closes the gap.

connected · lagging · dropped · Transitions, not levels · No MT4 host declared

05 MENU · ALERT SYSTEMS

Where a login came from, and what is still alive

The other four tabs of the same menu: the two that read logins out of the server's own log files, the board a NOC opens at three in the morning, and the door that puts all of it beside everything else the brokerage runs.

MT5 IP Geolocator `/server/geo/mt5`

Logins with the address they came from, and the ones from somewhere the account has never been. The screen opens by asking the trading gateway whether it keeps a login journal and printing its answer: today it is no, because the manager surface returns accounts, orders, positions, deals and symbols and none of them carries a login record. The route that does work is the Agent reading the login lines out of the server's own log directory, where the server itself writes them.

Agent log source · Gateway probed each load · Declared address ranges · Somewhere new

MT4 IP Geolocator `/server/geo/mt4`

The same screen and the same source for the other platform — the Agent tails an MT4 log directory exactly as it tails an MT5 one, and the login line is the same kind of line. The gateway half of the banner answers that there is no MT4 trading gateway on this installation.

Same source · No MT4 gateway

This platform ships **no geolocation database and makes no outbound call to one**, so it does not pretend to know which city an address is in. What it knows is exact: the network an address belongs to. An account that has used three networks for a year and appears on a fourth is a real event, stated as a fact. A brokerage that knows its own ranges names them, and those logins are labelled in the operator's own words. An account's first login is never flagged as new — flagging every one of them trains a desk to ignore the column.

Watch Dog `/server/watchdog`

Is everything alive. The verdict is the first thing on the screen, in words; the counts sit under it; the table is sorted worst-first and nothing else; and every row carries when it was last heard from rather than only what it last said. The patrol behind it runs on **this Hub's clock**, not the broker's machine — because the most important alert this module can send is that a machine has stopped talking, and that machine cannot raise it.

Verdict in words · Worst first · Hub-side patrol · Agent + service inventory

Metrics Export `/server/metrics`

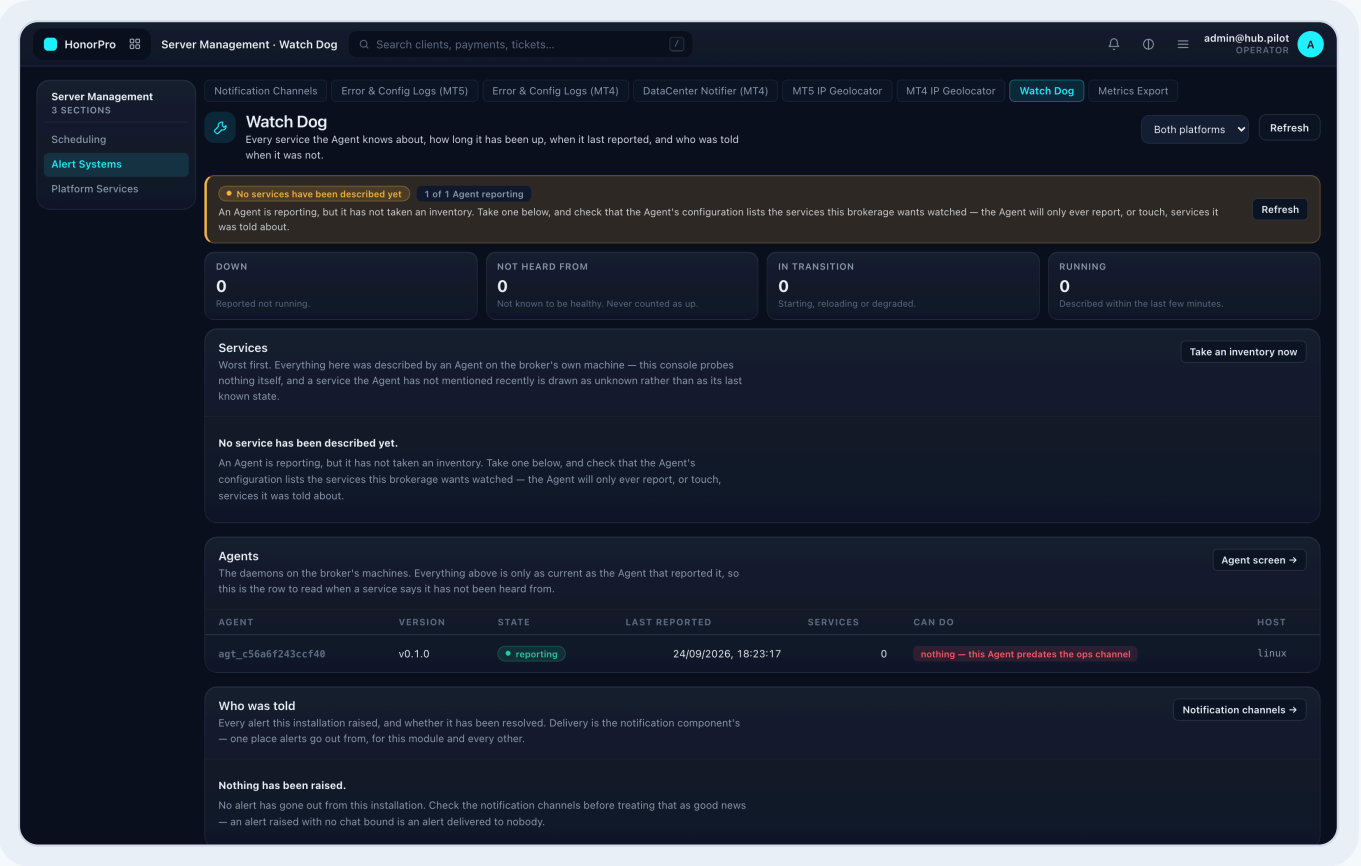
A brokerage already runs monitoring, with a screen carrying every other service they own. This door puts these numbers beside them: Prometheus text, scraped by Zabbix just as happily, behind one shared token presented as a header and never in a URL. The screen is three things — whether the door is open, the snippet to paste into their own monitoring, and the live output, so somebody can see what they are about to scrape before wiring it up.

Prometheus text · Zabbix HTTP agent · Bearer or header token · No session, it is not a person

06 IS EVERYTHING ALIVE

Four counts, and one of them is "not heard from"

The Watch Dog is the page a NOC opens at three in the morning. The verdict is a band in words, the counts sit under it, and the fourth count exists because a service nobody has heard from is a different thing from one that is down, and a different thing again from one that is running.



Alert Systems · Watch Dog. An Agent is reporting on this pilot but has taken no inventory, so nothing is being watched — and the band says that, rather than four zeros reading as a clean board.

Nothing here is probed from the Hub

Every row was described by the Agent on the broker's own machine, and a service it has not mentioned recently is drawn as unknown rather than as its last known state. That is why the Agents table sits under the services: it is the row to read when a service says it has not been heard from.

Who was told

Under the board is every alert this installation raised and whether it has been resolved. "Nothing has been raised" carries the warning that belongs with it — check the notification channels before treating that as good news, because an alert with no chat bound is an alert delivered to nobody.

07 MENU · PLATFORM SERVICES

The trading server's own administration

Six of these tabs are the gateway's own admin screens, reached with the session this console already holds. They are embedded rather than rewritten: a second implementation of the client book would be a second thing to keep true.

Accounts `/terminal/clients`

The client book on this gateway: who is registered, their group, leverage, balance and standing.

Desk permission: read

Orders `/terminal/orders`

What is working on the server right now — the live order book as the trading server itself holds it.

Desk permission: read

History `/terminal/history`

Closed deals as the server records them — the record every detector and every report elsewhere in this console is ultimately reading.

Desk permission: read

Groups `/terminal/groups`

The server's own group tree — what decides a client's trading conditions, symbol by symbol.

Desk permission: read

Audit log `/terminal/audit`

The trading server's own record of who changed what on it. The Hub's audit trail covers the Hub; this one covers the server, and nothing else here does.

Desk permission: read

Operators `/terminal/users`

The seats on the trading server itself — the manager and dealer logins it authenticates.

Desk permission: users

The permission on these six is the **trading server's**, not this console's; the embedded screen reports it once connected, and until then every tab is drawn.

08 MENU · PLATFORM SERVICES

The lines this installation holds, and the daemon that carries them

The last two tabs are the Hub's own, not the gateway's: every service this installation talks to, and the one piece of the platform that runs on the broker's machine.

Connections `/connections`

Every plugin and service this installation talks to on a desk of its own: the MT5 manager, the bridges and the copy engine the gateway holds and this screen administers, beside the desks it only links out to — the liquidity cover, the price feed, the CRM. Reachability is read honestly: any HTTP answer at all means something is listening, and only silence means down.

A secret is write-only. The password on a manager and the token on a bridge are never sent to this console and never rendered; the field is blank with a note saying the stored one is unchanged, and it is only put in the request when an operator actually types a new one. Editing is per kind — a manager takes a host, a login and a password, a bridge or the copy engine a URL and a token, and a read-only line takes nothing at all but carries a door to the module that does configure it.

Manager · bridges · copy engine | Read-only kinds link out | Secrets write-only | Install-level, not licensed

Agent `/agent`

The daemon on the broker's host, and the three things somebody needs in the order they need them: the enrollment token an Agent is installed with, the two lines to type on the gateway host with this Hub's address and the live token already in them, and every Agent that has enrolled — its version, when it last reported, and what it said about itself and the gateway beside it.

The token is one-time. Enrolling burns it, so a box that already has one shows no token, and that is the correct, healthy state; minting a new one invalidates the previous immediately, which is the whole point of the button. The Agent keeps a key the Hub stores only as a hash. A brokerage normally runs one Agent, and more than one is allowed: a second trading host is a second Agent, not a second tenant.

One-time enrollment | Key stored as a hash | Version and last report | Several hosts allowed

An Agent that predates the ops channel **declares nothing** rather than declaring an empty list, and the screens treat those as different answers: "never said" is not "said it can do nothing". A restart aimed at such an Agent is refused with the version it is running and the instruction to update it.

Everything this Hub sends the brokerage's gateway travels down the Agent's socket — the licence, the components that were switched on, the white label, the watchdog's thresholds — which is why "is the Agent reporting" is the question behind half the other screens in this console.

What is **not** in this menu is as deliberate as what is. Symbols is Group & Symbol Manager's Symbol Book, on the same data and a better screen; support tickets belong to the CRM, which owns tickets; the gateway's own settings screen is the Connections tab beside this one. Each would otherwise be a second door onto a surface that already has one.

09 HOW IT CONNECTS

What each screen reads, and what it is allowed to write

Only two tabs in this module can change anything on the broker's machine, and both do it by queueing a job the Agent chooses to execute. The rest read what the Agent has already volunteered.

TAB	READS	WRITES
Scheduled Restarts (MT5, MT4)	The service inventory the Agent took, and this installation's market hours	A schedule, and a run record — the restart itself is a job the Agent executes
Notification Channels	The bound chats and the alert events	A channel binding; outbound messages on delivery
Error & Config Logs	Log lines the Agent delivered, matched against this installation's patterns	The patterns, and an alert on a match
DataCenter Notifier	The MT4 ring as the Agent reads it off that host	A transition record, and an alert on a change
IP Geolocator	Login lines from the Agent; the gateway is asked whether it has a journal of its own	The observation, the account's network history, and the brokerage's own declared ranges
Watch Dog	Agent heartbeats and the service inventory — nothing is probed from here	Opens and closes incidents; raises the alert nobody on the broker's side could
Metrics Export	Everything above, as counters and gauges	Nothing; it is a scrape
The six desk tabs	The gateway's own admin screens, with the gateway's own permissions	Whatever that seat is allowed to change on the trading server
Connections, Agent	This installation's own service lines and enrolled Agents	Connection settings; the enrollment token

Licensing

Server Management is install-level plumbing rather than a licensed component: a brokerage administers the machine it is running on. Platform Services additionally needs the console session's grant on the gateway, and each of its six embedded screens then answers to the trading server's own permission for that seat.

When the Agent is not there

Every screen in this module opens with the same banner, composed on the server so six screens cannot answer it six ways: green when something is genuinely watching, amber when it is watching but stale, red when nothing is. It does not disappear on the good path — a NOC that only sees the line when something is wrong learns nothing from its absence.

Requests that reach the broker's machine are queued as jobs with two deadlines: one for the Agent to claim the job, and one for it to answer. A job nobody claims expires saying the Agent is not connected; a job claimed and never answered fails saying it may have been restarted while the work was running. Neither is left looking like success.