

Meridian Aegis

Guía de instalación y operación

Version 2.1.0 · https://meridianrp.es/aegis/ · Anticheat server-authoritative para FiveM

Índice

1. 1. Requisitos
2. 2. Instalación
3. 3. Modos de funcionamiento
4. 4. Filosofía de castigo
5. 5. El panel /aegis
6. 6. EventGuard, contratos de eventos
7. 7. EntityGuard
8. 8. EconomyGuard
9. 9. Sensores de cliente, combate y otros detectores
10. 10. Comandos
11. 11. Configuración (`shared/config.lua`)
12. 12. Licencia y control remoto
13. 13. Privacidad
14. 14. Problemas frecuentes y FAQ
15. 15. Exports (API de servidor)
16. 16. Soporte

FiveM Security & Anti-Cheat Platform, versión 2.1.0

Aegis es una plataforma de seguridad server-authoritative para FiveM. El cliente solo reporta; todas las decisiones (riesgo, kick, ban, bloqueo) se toman en el servidor. Este documento describe lo que el recurso hace de verdad, cómo instalarlo y cómo operarlo. Si algo de lo que lees aquí no coincide con lo que ves en tu servidor, el servidor manda: contacta con soporte.

English version: `DOCUMENTATION.md` (también en https://meridianrp.es/aegis/docs).

---

1. Requisitos

Obligatorios:

- FXServer con un artifact reciente (el build recomendado por Cfx.re).
- OneSync activado (`set onesync on`). Aegis es server-authoritative y necesita OneSync para leer entidades y jugadores.
- Lua 5.4. El manifest ya declara `lua54 'yes'`; no hay que hacer nada.

Opcionales (Aegis funciona sin ellos):

- `oxmysql`: persistencia en base de datos de bans, señales, acciones y auditoría de staff. Sin él, los bans se guardan en `data/bans.json` dentro del recurso y sobreviven igualmente a los reinicios.
- `screenshot-basic` (recurso oficial de Cfx): capturas de pantalla como evidencia de los incidentes. Necesita una carpeta `.tmp` con permisos de escritura dentro del recurso `screenshot-basic`. Aegis intenta crearla al arrancar y avisa por consola si no puede.
- Framework: `qbx_core`, `qb-core`, `es_extended` o standalone. La detección es automática por el estado del recurso y se repite si el framework arranca después que Aegis.

2. Instalación

1. Descomprime el paquete en tu carpeta `resources`. **No renombres la carpeta**: tiene que llamarse `meridian_aegis`. La ruta del sidecar `@meridian_aegis/bootstrap/resource.lua` y la vinculación de la licencia dependen del nombre.
2. Comprueba que la carpeta `data/` existe dentro del recurso y tiene permisos de escritura. Viene en el zip con un `README.txt`. Aegis escribe ahí `serverid.txt`, `settings.json`, `contracts.json`, `bans.json` y `aegis-support.json`.
3. No hay nada que configurar para la licencia. El acceso y la duración los gestionan Tebex y el Asset Escrow de Cfx.re. Opcionalmente, vincula el servidor a tu portal de comprador: al arrancar, la consola muestra `Código de portal: XXXX-XXXX`; pégalo en https://meridianrp.es/aegis/ → "Vincular servidor" (ver 2.1). Aegis protege exactamente igual lo vincules o no.

4. Da acceso al panel a tu grupo de administración:

```
add_ace group.admin aegis.admin allow
```

5. Orden de arranque recomendado en `server.cfg`: Aegis funciona en cualquier orden (vuelve a detectar el framework y la base de datos cuando arrancan), pero este orden evita avisos al arrancar.

```
ensure oxmysql            # opcional
ensure qbx_core           # o qb-core / es_extended, si usas uno
ensure screenshot-basic   # opcional
ensure meridian_aegis
```

6. Arranca el servidor. En consola deberías ver: y, unos segundos después, `licencia válida` si el servidor está vinculado al portal. Si no lo está, la consola muestra en su lugar `Servidor sin vincular · Código de portal: XXXX-XXXX`; es informativo, la protección ya está funcionando.

```
[Aegis:main] Meridian Aegis 2.1.0 listo · modo=shadow
```

2.1 El portal del comprador y el código de portal

No hay que escribir nada en tu servidor para que Aegis funcione. El recurso se entrega por Asset Escrow de Cfx.re y Tebex controla durante cuánto tiempo puede usarlo tu servidor.

El portal del comprador, https://meridianrp.es/aegis/ , muestra tu licencia (plan, días restantes, renovación), tus servidores vinculados, la versión publicada y las descargas. Entras con el id de transacción de Tebex (pestaña "Acabo de comprar") o con el código de licencia que te da el propio portal (`AEGIS-XXXX-XXXX-XXXX-XXXX`). Ese código es solo para el portal; nunca se configura en el servidor.

Para ver un servidor en el portal, vincúlalo: al arrancar, Aegis imprime en consola `Código de portal: XXXX-XXXX` (también con el comando `aegis_portal`). Pégalo en el portal → "Vincular servidor". El código se deriva de un hash de un id de servidor aleatorio que Aegis genera en el primer arranque y persiste en `data/serverid.txt` (FXServer oculta `sv_licenseKey` a los recursos, así que la licencia en sí nunca se lee ni se envía). Es estable para ese servidor mientras conserves `data/` y no revela nada de él; si borras `data/`, el id y el código cambian y tienes que vincular el servidor otra vez (ver 14). Una licencia vincula hasta **2 servidores** (producción y desarrollo); puedes desvincular uno desde el portal cuando quieras. Si los dos huecos están ocupados, un servidor vinculado que lleve **14 días** sin verse se libera solo al vincular uno nuevo. Los servidores vinculados reciben además los avisos de versión.

2.2 Permisos en detalle

- `aegis.admin` (ACE): abre el panel `/aegis` y desbloquea los comandos de administración desde el chat. Por defecto, quien lo tiene también es **inmune** al anticheat (ver 2.3).

- Los comandos de consola están registrados como restringidos. Desde la consola del servidor (incluida la de txAdmin) funcionan siempre. Para ejecutarlos desde el chat del juego, el jugador necesita el ACE `command.<nombre>` de ese comando **y** acceso al panel. La línea por defecto del `server.cfg` `add_ace group.admin command allow` ya cubre a `group.admin`; para otro grupo añade, por ejemplo, `add_ace group.mods command.aegis_status allow`. En la práctica, usa el panel: todo lo que hacen los comandos está ahí.
- Acceso al panel sin ACE: desde el panel, Ajustes, "Quién puede usar /aegis", el dueño puede dar acceso a jugadores concretos por nombre o identificador (`license:`, `discord:`, `fivem:`, `steam:`). La lista se guarda en el acto en `data/settings.json` y sobrevive a los reinicios.
- Sin ACE y sin estar en esa lista, `/aegis` responde "sin permiso" y no hace nada.

2.3 Inmunidad de admins

Los admins usan noclip, godmode, teleports y spawns como parte de su trabajo. Un jugador **inmune** se ignora por completo: sin señales, sin riesgo, sin kicks, sin bans, sin presupuestos de spawns ni explosiones, sin anti-spam de chat, sin contratos de EventGuard. Un jugador es inmune si se cumple cualquiera de estas condiciones (`config.admins`):

1. ACE `aegis.immune` (`add_ace group.admin aegis.immune allow`).
2. Puede abrir el panel (`aegis.admin`) y `admins.includePanelAdmins` está a true (por defecto).
3. Pertenece a un grupo del framework listado en `admins.groups` (por defecto `admin`, `superadmin`, `god`).
4. Su identificador está en `admins.identifiers`.
5. Un hook del servidor `Aegis.ImmuneHook` devuelve true.

La lista se reevalúa cada `admins.cacheSeconds` (60 s). En el panel, Ajustes, "Admins inmunes" se editan los puntos 2 y 3.

2.4 Base de datos

Dos opciones, las dos válidas:

- Importa `aegis.sql` (InnoDB, utf8mb4) en tu base de datos antes de arrancar.
- No hagas nada: con `oxmysql` arrancado y `audit.persist = true` (por defecto), Aegis crea las tablas `aegis_bans`, `aegis_signals`, `aegis_actions` y `aegis_staff_audit` en el primer arranque.

Las filas antiguas se podan automáticamente al arrancar según `audit.retentionDays` (30 por defecto). Solo se escriben en la base de datos las señales con severidad igual o superior a `audit.persistMinSeverity` (50) o con prueba dura. Si tenías bans en `data/bans.json` antes de añadir `oxmysql`, se migran en el siguiente arranque.

3. Modos de funcionamiento

Aegis arranca en modo **Shadow**, que el panel llama **Test**.

Modo	Nombre en el panel	Qué hace
`shadow`	Test	Detecta, calcula riesgo, registra incidentes y evidencia, **aprende el tráfico legítimo de eventos para proponer contratos** y lo muestra todo en el panel. **No** kickea, banea ni bloquea nada.
`live`	Activo	Enforcement real: kicks, bans, bloqueo de eventos, presupuestos de entidades, anti-spam de chat.

Se cambia desde el panel (selector de modo arriba, y después "Guardar cambios" para que sobreviva al reinicio) o con `aegis_mode live`. Mientras el modo no sea `live`, la consola recuerda cada 5 minutos que Aegis solo observa, para que nadie deje el servidor sin proteger por error.

Cosas que fuerzan el comportamiento de Test aunque tú tengas otro modo:

- Un **kill switch** o un **modo forzado** firmados desde el backend de licencias, que solo se usan si aparece un bug grave en una versión. Lo verás en la barra de estado del panel y en consola.
- Si el backend de licencias no responde, Aegis sigue funcionando con una caché local firmada (ver 12). **Nunca** se apaga solo porque el backend esté caído.

En Test no se avisa a los jugadores. `shadowChat = false` por defecto; ponlo a true solo mientras calibras tú mismo, porque le dice al jugador lo que Aegis habría hecho.

Despliegue recomendado: 1 o 2 días en Test, revisar Incidentes, poner en "Registrar" los detectores ruidosos en Detecciones, y entonces pasar a Activo.

4. Filosofía de castigo

El cliente reporta, el servidor decide. Las reglas que importan:

- **Riesgo acumulado con decaimiento.** Cada señal suma una contribución ponderada que se reduce a la mitad cada `risk.halfLifeMs` (5 min). La prueba dura (replay, protocolo imposible) decae mucho más despacio (1 h).
- **La persistencia cuenta ventanas de 12 segundos, no ráfagas.** Un detector suma como máximo un paso de persistencia por `risk.persistWindowMs`. Treinta props al entrar en una casa o nueve clics en una tienda cuentan una vez. Solo un cheat mantenido durante minutos escala.
- **Correlación.** Señales de categorías distintas en la misma ventana se amplifican entre sí. Un detector no-hard por sí solo nunca pasa de **cuarentena**; un kick por riesgo necesita al menos `risk.kickMinCategories` (2) categorías distintas, o una acción Kick/Ban explícita en ese detector.
- **Tiers de riesgo** (`tiers`): 0 observar, 20 registrar, 40 bloquear, 60 cuarentena, 80 kick, 100 candidato a ban. Un candidato a ban va a la cola de **Revisión**, nunca a un ban automático.

- **Auto-ban de fábrica**: solo `honeypot.trap` (un evento trampa que ningún recurso legítimo dispara). `autoBan.enabled` se puede apagar en Ajustes; entonces todo pasa por la cola de revisión.
- **Auto-kick de fábrica** (`autoKick`, necesita `minCount = 2` muestras sostenidas de la misma señal): `sensor.noclip`, `sensor.invisible`, `sensor.super_jump`, `sensor.weapon_mod`, `sensor.magic_bullet`, `sensor.no_recoil`, `sensor.vehicle_fly`, `sensor.speed`. Los detectores que dependen del entorno del servidor (godmode, godmode de vehículo, munición infinita, mass spawn, explosiones) solo suman riesgo por defecto.
- **Acción por detector** desde el panel (Detecciones): Ignorar, Registrar, Kick o Ban para cada detector. Se guarda en `detectorActions` y se persiste con "Guardar cambios". Kick y Ban puestos aquí actúan a la primera muestra.
- **Cola de revisión con evidencia**: cada incidente serio congela un caso (telemetría antes y después, señales, replay 2D, capturas si las hay). El staff confirma el ban con la evidencia o lo descarta, opcionalmente marcándolo como falso positivo, lo que resta riesgo y alimenta las métricas de calidad del detector.
- **Bans por identificadores estables**: `license`, `discord`, `fivem`, `xbl`, `live`. Nunca por IP y nunca por Steam solo.
- **Cuarentena**: restringe recompensas, creación de entidades y economía sin expulsar. Se levanta desde la ficha del jugador.

4.1 Anuncio público

Cuando Aegis expulsa o banea (automático o desde el panel) en modo Activo, lo anuncia en el chat, en el idioma de Aegis:

```
[AEGIS AC] Jugador ha sido baneado por Motivo
```

Configurable en Ajustes, "Anuncio público", y en `config.announce`: activado, audiencia (`all` todo el servidor, `staff` solo staff, `none`), mostrar el motivo o no, prefijo (máximo 24 caracteres), color del prefijo, nombre del personaje o nombre de FiveM, comillas. En Test no se anuncia nada.

4.2 Qué ve el jugador

- Kick o ban: pantalla de desconexión con un motivo humano, en el idioma de Aegis (es, en, pt).
- Al reconectar estando baneado: tarjeta de conexión con el motivo y un botón **Apelar** que abre `enforcement.appealUrl` (Ajustes, "Discord de apelación"). Pon tu propia invitación de Discord; la de fábrica es un marcador.

5. El panel /aegis

Escribe `/aegis` en el chat. Requiere `aegis.admin` o estar en la lista de staff (2.2). Cierre de emergencia si el panel se queda con el foco del ratón: `/aegis_close`.

Barra superior: selector de modo (Test, Activo), botón **Guardar cambios** (escribe `data/settings.json`), idioma (ES, EN, PT; también `aegis_lang en`). Barra de estado: modo, detectores, jugadores conectados, eventos protegidos, puntuación de salud, y un aviso cuando el kill switch o un modo forzado mandan sobre el modo local.

Secciones:

Sección	Qué encuentras
Incidentes	Agrupados por jugador con historial, riesgo y el detector que saltó. Espectar en vivo, abrir el replay, ir al jugador. Se pueden mostrar los archivados.
Jugadores	Todos los conectados con estado (limpio, vigilado, en alerta). Abre la ficha del jugador: riesgo, señales, espectar, kick, ban, cuarentena, soltar.
Evidencia	Casos guardados con replay 2D (trayectoria, disparos, marcadores) y capturas.
Baneados	Fecha, staff, motivo, evidencia. Desbanear con un clic.
Revisión	Candidatos a ban pendientes de confirmar, con su caso. Confirmar, descartar, descartar como falso positivo, resolver apelaciones.
Detecciones	Cada detector con su acción actual: Ignorar, Registrar, Kick, Ban. Cada uno explica qué detecta.
Ajustes	Modo, idioma, interruptor de auto-ban, Discord de apelación, quién puede usar /aegis, anuncio público, admins inmunes, política de armas (Legal, Avisar, Kick, Ban por arma; requiere `GrantWeapon` en tu inventario).
Entidades (avanzado)	Captura en vivo de las entidades que crean los jugadores (quién, modelo, procedencia). Borrar una entidad o poner en cuarentena a su creador.
Escáner de recursos (avanzado)	ResourceShield: escanea todos los recursos, lista los eventos explotables con sus razones, botones Proteger y Desproteger, botón de reinicio, editor de contrato por evento.
EventGuard (avanzado)	Eventos protegidos y estadísticas de sus contratos.
Economía (avanzado)	Circuit breakers y anomalías; rearmar un breaker.
Calidad de detección (avanzado)	Por detector: señales, confirmados, descartados, falsos positivos, tiempo p95. Restaurar un detector que se degradó solo a shadow.
Capacidades (avanzado)	Catálogo de tickets de capacidad y catálogo de armas.
Asistente (avanzado)	Lista de pasos de puesta en marcha.

Sección	Qué encuentras
Diagnóstico (avanzado)	Chequeos del entorno (artifact, OneSync, base de datos, framework, inventario, licencia, reloj, permisos, orden de recursos, conflictos conocidos). Exportar un paquete de soporte sin secretos.
Salud (avanzado)	Salud del runtime y contadores.

Qué persiste "Guardar cambios": modo, idioma, acción por detector, detectores desactivados, política de armas, opt-in del reto anti-tamper, interruptor de auto-ban, URL de apelación, anuncio público, admins inmunes, lista de staff. La lista de staff se guarda en el momento en que la editas.

6. EventGuard, contratos de eventos

La mayoría de exploits de FiveM no son "cheats": son eventos legítimos del servidor disparados con argumentos falsos (`TriggerServerEvent('job:reward', 999999)`). EventGuard te permite declarar cómo es una llamada válida y rechaza todo lo demás, con evidencia.

Un contrato (policy) puede incluir:

Campo	Significado
`rate = { max, window }`	Máximo de llamadas por ventana (ms) y jugador.
`args = { { index, type, min, max, equals, oneOf } }`	Esquema de cada argumento; `equals` puede referirse al contexto de sesión (`'session.deliveryId'`).
`distance = { source, max }`	El jugador debe estar a menos de `max` metros de un punto (un campo de la sesión o unas coordenadas).
`requireSession = { type, stage }`	Debe existir una sesión de seguridad de ese tipo (y fase) para el jugador.
`operation`, `completesOperation`	Nombra una operación para que solo pueda completarse una vez por sesión.
`antiReplay = true`	El primer argumento debe ser un nonce emitido por `IssueNonce`; cada nonce es de un solo uso.
`signed = true`	Trigger firmado (ver abajo): el evento solo pasa si lo disparó el propio cliente del recurso a través del sidecar de Aegis. Solo contratos manuales.
`permissions = { ace, job }`	Quien llama debe tener ese ACE o ese job.
`direction`	Dirección esperada (cliente a servidor) para cazar eventos internos falsificados.
`advanceStage`	Pasa la sesión a otra fase cuando el evento es válido.

Campo	Significado
`action`, `onViolation`	Sobrescriben la acción ante una violación.

Las violaciones generan señales `protocol.*` (`rate`, `schema`, `distance`, `session`, `session_owner`, `session_expired`, `state`, `replay`, `duplicate`, `permission`, `direction`, `generic`). En Test o con kill switch **nunca se bloquea un evento**: la violación se registra y el handler se ejecuta.

6.1 Opción A, proteger desde el panel (sin código)

Escáner de recursos, elige un recurso, **Proteger**. Aegis:

1. Escanea el recurso (Lua, JavaScript, C#) y crea un contrato automático seguro para cada evento alcanzable desde el cliente con severidad MEDIA o superior. Los contratos automáticos solo fijan un **rate** razonable; sesión, distancia, permisos y "derivar el valor en el servidor" aparecen como sugerencias que activas a mano en la ficha del recurso.
2. Añade una línea marcada y reversible al `fxmanifest.lua` del recurso: con copia de seguridad y checksum. Ese **sidecar** se carga antes que los scripts del recurso y envuelve `RegisterNetEvent`, `AddEventHandler` y `lib.callback.register`. En los eventos con contrato consulta a Aegis antes de ejecutar el handler. Fail-open: si Aegis está parado o falla, el handler se ejecuta.

   ```
   shared_script '@meridian_aegis/bootstrap/resource.lua'
   ```
3. Muestra el Security Diff (puntuación antes y después).

Estados en el panel: Sin proteger, Pendiente de reinicio (manifest escrito, sidecar aún no cargado: ejecuta `refresh; ensure <recurso>`), Protegido y activo (el sidecar ha saludado). Para que funcione el botón Reiniciar del panel, añade:

```
add_ace resource.meridian_aegis command.refresh allow
add_ace resource.meridian_aegis command.ensure allow
```

Desproteger retira los contratos y la línea del manifest. Los contratos viven en `data/contracts.json` y se recargan al arrancar; editar uno en el panel recarga el sidecar en caliente. Los recursos con Asset Escrow no se modifican nunca: sus contratos quedan en EventGuard y se integran con `ProtectEvent` o `RegisterProtectedEvent` en código.

6.2 Opción B, SDK en tus propios recursos

```
-- Declara el contrato y registra el handler en una sola llamada.
exports.meridian_aegis:RegisterProtectedEvent('delivery:complete', {
    rate = { max = 1, window = 3000 },
    requireSession = { type = 'delivery', stage = 'at_destination' },
    distance = { source = 'session.destination', max = 8.0 },
    args = { { index = 1, type = 'string', equals = 'session.deliveryId' } },
    antiReplay = true,
    operation = 'delivery', completesOperation = 'delivery',
}, function(src, deliveryId)
    -- Solo se ejecuta si Aegis lo aprobó. La recompensa la pone el SERVIDOR, nunca el cliente.
    local reward = ActiveDeliveries[src].reward
    exports.qbx_core:AddMoney(src, 'bank', reward)
end)

-- Antes, cuando empieza la entrega:
local id = exports.meridian_aegis:StartSecuritySession(src, 'delivery', {
    stage = 'in_transit', destination = coords, deliveryId = 'A32', reward = 450,
})
exports.meridian_aegis:SetSessionStage(id, 'at_destination')

-- Anti-replay: emite un nonce en el servidor, pásaselo a tu cliente (callback),
-- y el cliente lo envía como PRIMER argumento del evento protegido.
local nonce = exports.meridian_aegis:IssueNonce(src, 'delivery')
```

Si solo quieres validar sin cambiar tu handler, llama a `ProtectEvent(name, policy)` una vez y a `CheckEvent(source, name, ctx)` dentro del handler; `ctx.args`, `ctx.nonce` y `ctx.playerCoords` son lo que tú pases.

7. EntityGuard

Presupuestos server-authoritative de creación de entidades por jugador y por servidor (`entityguard.limits`), detección de ráfagas (`entityguard.burst`, con una gracia de conexión de 45 s para que el streaming al entrar nunca salte), tipos de explosión prohibidos y límites de spawn de población mission (`entityguard.spawnLimits`, ped 12 / vehículo 12 en 6 s, objeto 30 en 4 s). Las explosiones legítimas del juego (vehículos, surtidores, fuegos artificiales, etc.) se excluyen con `entityguard.explosion.ignoreTypes`.

Menús inyectados: `sensor.injection`. El cliente de Aegis fotografía su propio entorno de scripts al arrancar (funciones globales como `TriggerServerEvent`, `GetEntityHealth`, la lista de globales, hooks de debug) y lo compara cada `sensors.integrity.intervalMs`. Caza a los menús que se inyectan en el **entorno Lua** y sustituyen esas referencias.

Alcance honesto (importante): la integridad de entorno **NO** detecta a los menús premium de nivel motor (redEngine, Eulen, Tiago y similares). Esos parchean las nativas del juego **en memoria (C++)** y no cambian la referencia Lua, así que la foto del entorno queda idéntica a limpio. A esos cheats se les caza por **comportamiento en el servidor** (godmode, teleport, spawn masivo, arma sin procedencia, velocidad, invisibilidad), que es la **primera línea** de Aegis.

`sensor.injection` es una capa más, no la principal, y no debes venderla ni configurarla como «detector de todos los menús». `aegis_integrity_dump <id>` imprime la diferencia del entorno del cliente para construir firmas de un menú de nivel Lua concreto.

Cobertura en todos los recursos: un menú solo puede enganchar el entorno al que apunta, así que el mismo vigilante va también dentro del sidecar (`bootstrap/resource.lua`). Cada recurso protegido con `aegis_protect <res> apply` o `aegis_protect_all` ejecuta en cliente una copia que fotografía *su propio* entorno y reporta a través de Aegis (`IntegrityReport`), con respaldo directo al servidor si Aegis no está cargado. La evidencia dice en qué recurso se vio el hook (`GetEntityHealth@es_extended`) y el servidor cuenta el mismo hallazgo una vez por jugador cada 5 minutos, así que 150 sidecars viendo el mismo hook son una señal, no 150. Todo el sidecar va en `pcall` y falla abierto: un Aegis roto nunca rompe un recurso protegido.

Triggers firmados (`signed = true`): el mismo evento, por ejemplo `esx_drugs:pickedUpPoppy`, es legítimo cuando lo dispara el script de droga tras la animación y un exploit cuando lo dispara un ejecutor en bucle. Aegis los distingue por el origen. Al entrar, el servidor entrega a cada cliente un secreto de sesión; el sidecar de cliente que vive en cada recurso protegido envuelve `TriggerServerEvent` en ese recurso y, para los eventos cuyo contrato tenga `signed = true`, añade `contador:HMAC(secreto, recurso|evento|contador)` como último argumento. El sidecar de servidor lo verifica y lo quita antes de ejecutar el handler real, así que el recurso nunca ve el token. Un ejecutor dispara desde su propio entorno, sin el wrapper: el evento llega sin firma (o con contador repetido) y se bloquea como `protocol.unsigned`, prueba dura, severidad 90. Si el trigger sale de dentro del entorno del recurso pero no de uno de sus ficheros (un chunk inyectado), el cliente además reporta `sensor.caller`, señal blanda, porque hay recursos legítimos que compilan código con `load()`.

Se activa por evento desde la pestaña Scanner (casilla "Trigger firmado") o con `SetContract(res, evento, { rate = ..., signed = true })`. Requisitos: el recurso tiene que estar protegido (`aegis_protect <res> apply` o `aegis_protect_all`, y reinicio después) para que cargue el sidecar de cliente, y el evento tiene que dispararse desde el cliente Lua del recurso. No lo actives en eventos que se disparan desde clientes JavaScript o desde callbacks NUI que no pasan por Lua, ni en eventos que otros recursos disparan en nombre de este: llegarían sin firma. En modo Test los triggers sin firma pasan y se registran como "bloquearía", así que puedes activarlo en un servidor lleno y mirar el panel antes de pasar a Activo. Límite honesto: el secreto vive en el cliente, así que un menú que lo extraiga puede firmar; por eso el sensor de integridad y los contratos de servidor siguen ahí. El trigger firmado mata el trigger básico, que es lo que hace casi todo el que usa un ejecutor.

Protocolo de eventos polimórfico (`cloak = true`, **EXPERIMENTAL, apagado por defecto**): evolución de los triggers firmados. En vez de firmar el evento, **oculta su nombre**: el sidecar reescribe el `TriggerServerEvent` de los eventos cloak a un envío por `aegis:rpc` con un token que **rota por jugador y por época (15 s)**. Un event logger ve siempre `aegis:rpc` con un token opaco distinto, así que no puede saber cuál es «cosechar», y lo que capture está muerto en la siguiente época/sesión y para cualquier otro jugador: «loguear y repetir» deja de funcionar. Incluye eventos **señuelo** (un volcador de mapa que reenvía todo dispara uno, prueba dura) y contador anti-replay. Va **apagado por defecto** (`cloak.enabled = false` en `shared/config.lua`); es una capa

avanzada que **debes validar en tu propio servidor** antes de activarla, con las mismas reglas que los triggers firmados (nunca en eventos disparados desde JS/NUI o desde otros recursos). Límite honesto idéntico: el secreto vive en el cliente; por eso se combina con los detectores de comportamiento.

Señales: `entity.burst`, `entity.cage` (varios objetos creados por el cliente alrededor de otro jugador en pocos segundos, la jaula típica de mod menu; en modo Activo se cancela el objeto que la completa y se borran los anteriores, `entityguard.cage`), `entity.explosion`, `entity.forbidden`, `entity.unattributed`, `sensor.mass_spawn`. Por defecto suman riesgo y aparecen en el panel; súbelas en Detecciones cuando hayas declarado tus propios spawns. `deleteOnMassSpawn` (borrar la entidad en modo Activo) es opt-in y viene apagado.

Un recurso que crea entidades de forma legítima (garaje, atraco, concesionario) las declara y queda exento:

```
exports.meridian_aegis:EntityExpect(GetCurrentResourceName(), 'vehicle', 1, 5000)
exports.meridian_aegis:EntityExpect(GetCurrentResourceName(), 'ped', 8, 10000)
```

8. EconomyGuard

Engancha el AddMoney del framework y vigila las ganancias de gameplay por jugador (`economy.player`, por defecto 500k en 5 min, 300 items en 5 min) y por recurso frente a su propia línea base (`economy.resource.anomalyFactor` 8x). Los motivos que encajan con `economy.ignoreReasons` (bank, transfer, sell, society, casino, paycheck, etc., por subcadena) no cuentan como ganancia. Ante una anomalía, un **circuit breaker** granular pausa solo esa operación de ese recurso durante `economy.breaker.ttl` (60 s); se rearma desde Economía o con `aegis_breakers`.

```
local r = exports.meridian_aegis:GuardReward({ resource = 'delivery', operation = 'pay', player =
src, amount = reward })
if not r.allow then return end
```

9. Sensores de cliente, combate y otros detectores

Los sensores de cliente (`sensor.*`) se tratan como pistas de baja confianza; no se reporta nada hasta que el jugador está cargado del todo (`sensors.loadedGraceMs`, 20 s). Los invariantes derivados en servidor pesan más: `invariant.teleport` (más de 200 m/s) y `invariant.speed` (velocidad a pie sostenida, tres muestras consecutivas). Las gracias de spawn, respawn y salida de vehículo van incorporadas.

Combat Intelligence corre en el servidor a partir de los eventos de daño reales: `sensor.godmode` (la vida nunca baja bajo impactos confirmados), `combat.damage_mod` (daño de arma muy por encima del catálogo), `combat.firerate` (cadencia imposible), `aim.snap`, `aim.reaction`, `aim.tracking` (perfil

acumulado sobre al menos 10 disparos a jugadores reales; las dianas NPC se ignoran). Los detectores de aim nunca castigan solos; correlacionan.

Otros detectores: `weapon.forbidden` y `weapon.unregistered` (política de armas más la procedencia de `GrantWeapon`), `chat.spam` (bloquea el flood en servidor), `honeypot.trap`, `integrity.no_heartbeat` (reto anti-tamper, opt-in con `sensors.challenge.enabled`, apagado por defecto porque los reinicios de recursos y las cargas largas pueden retrasar la respuesta del cliente).

Calidad de detección: si la tasa de falsos positivos de un detector supera `quality.fpRateDemote` (25 %) tras `quality.minSamples` señales, se degrada solo a shadow y el panel te avisa. Se restaura desde Calidad de detección o con `aegis_restore`.

10. Comandos

Todos los comandos de servidor requieren consola o `aegis.admin` (o la lista de staff), más el ACE `command.<nombre>` cuando se escriben desde el chat. `aegis_aimtest` y `aegis_selftest` son **herramientas de prueba**, no para producción.

Comando	Qué hace
`aegis_status`	Versión, modo, número de detectores, sesiones y contadores de señales y acciones.
`aegis_mode <shadow, live>`	Cambia el modo: `shadow` = Test (detecta y aprende, nunca castiga), `live` = Activo. Guarda desde el panel para persistirlo.
`aegis_portal`	Muestra el código de portal de este servidor (para vincularlo en https://meridianrp.es/aegis/).
`aegis_lang <es, en, pt>`	Cambia el idioma de consola y de los mensajes a jugadores.
`aegis_risk <playerId>`	Imprime el riesgo actual de un jugador.
`aegis_scan [recurso]`	Escaneo ResourceShield de un recurso o de todo el servidor (puntuación, cobertura, recuentos).
`aegis_plan <recurso>`	Propone contratos para los eventos sensibles de un recurso.
`aegis_protect <recurso> [apply]`	Sin `apply`, muestra el diff del manifest sin escribir; con `apply`, contratos más sidecar.
`aegis_protect_all [escrow]`	Instala el sidecar en todos los recursos arrancados (salta los escrow salvo que pongas `escrow`). Reinicia el servidor después. Se deshace por recurso con `aegis_unprotect`.
`aegis_forensics <id> on/off`	**Modo forense (research).** Vuelca a consola y a `data/forensics.log`, con timestamp, todo lo que hace ese jugador (funciones enganchadas, globales nuevas, eventos que dispara, entidades, cada señal). Para analizar un cheat en tu server de pruebas y sacarle firmas. Solo staff; por defecto no captura a nadie.

Comando	Qué hace
`aegis_unprotect <recurso>`	Retira contratos y la línea del sidecar.
`aegis_learn`	Lista los contratos propuestos a partir del tráfico observado en modo Test.
`aegis_breakers`	Lista los circuit breakers de economía activos.
`aegis_cases`	Lista los últimos 20 casos de evidencia.
`aegis_case <id>`	Imprime los marcadores del replay de un caso.
`aegis_review`	Lista los candidatos a ban pendientes en la cola de revisión.
`aegis_ban <reviewId> [motivo]`	Confirma una revisión como ban, con su evidencia.
`aegis_dismiss <reviewId> [fp]`	Descarta una revisión; con `fp` la marca como falso positivo.
`aegis_quality`	Tabla de calidad por detector (señales, confirmados, descartados, tasa de FP, p95).
`aegis_fp <playerId> <detector> [suppress]`	Marca un falso positivo a mano y retira su riesgo.
`aegis_restore <detector>`	Vuelve a habilitar un detector degradado a shadow.
`aegis_corpus`	Ejecuta el corpus de regresión de falsos positivos.
`aegis_caps`	Imprime el catálogo de tickets de capacidad.
`aegis_capture <on, off>`	Enciende o apaga la captura en vivo de entidades (también en Entidades).
`aegis_report`	Imprime el Aegis Security Report (resumen de lo que Aegis vio e hizo).
`aegis_setup`	Imprime la lista de pasos del asistente de puesta en marcha.
`aegis_diagnostics`	Ejecuta los chequeos del entorno.
`aegis_support`	Escribe `data/aegis-support.json` (sin secretos) para enviarlo a soporte.
`aegis_aimtest <on, off>`	Herramienta de prueba: permite que el análisis de aim cuente dianas NPC. Apagado en producción.
`aegis_selftest`	Herramienta de prueba: dispara tres señales sintéticas para ver reaccionar el pipeline.
`/aegis` (cliente)	Abre el panel de control.
`/aegis_close` (cliente)	Cierra el panel a la fuerza si se queda con el foco.
`/aegisunspec` (cliente)	Deja de espectar y devuelve al staff a su posición.

11. Configuración (`shared/config.lua`)

`shared/config.lua` se carga como **datos**, no como script. Aegis lee el fichero y lo evalúa en un sandbox sin natives, sin hilos, sin HTTP y sin acceso al resto del recurso. Solo puede declarar la tabla `Aegis.Config`. Si contiene cualquier otra cosa (un `CreateThread`, una llamada a `exports`, `os.`, `_G`, etc.) Aegis se niega a arrancar e imprime un error claro: `[Aegis] CONFIG RECHAZADA: shared/config.lua contiene código no permitido (...)`. Corrige el fichero y reinicia.

Todo lo de abajo son valores por defecto; el panel sobrescribe varios en caliente y los persiste en `data/settings.json`, que se aplica encima de `config.lua` al arrancar.

Sección	Claves	Qué controla
`mode`	`shadow`, `live`	Modo de arranque. Sale de fábrica en shadow (Test).
`shadowChat`	booleano	Avisar al jugador por chat de lo que Test habría hecho. Apagado.
`locale`	`es`, `en`, `pt`	Idioma de consola, chat y mensajes de desconexión.
`risk`	`max`, `halfLifeMs`, `hardHalfLifeMs`, `pruneBelow`, `correlation`, `persistenceStep`, `persistenceCap`, `persistWindowMs`, `kickMinCategories`, `noPersistence`	Motor de riesgo: decaimiento, ventanas de persistencia, requisito de correlación. `pruneBelow` (0.5) descarta las contribuciones cuyo peso decaído baja de ahí; `correlation = { windowMs, perExtraCategory, maxBonus }` amplifica el riesgo total por cada categoría extra de señal vista dentro de la ventana (2 min, +12 % cada una, tope +60 %).
`tiers`, `tierAction`	umbrales	Mapa riesgo a acción. El candidato a ban siempre va a revisión del staff.
`banCandidateThreshold`	número	Riesgo (100) a partir del cual un jugador pasa a candidato a ban y entra en la cola de revisión; nunca un ban automático.
`admins`	`enabled`, `ace`, `includePanelAdmins`, `groups`, `identifiers`, `cacheSeconds`	Reglas de inmunidad (2.3).
`staff`	`identifiers`	Lista inicial de jugadores que pueden abrir /aegis; el panel la mantiene.
`announce`	`enabled`, `audience`, `showReason`, `prefix`, `color`, `useCharacterName`, `quotes`	Anuncio público de sanciones (4.1).

Sección	Claves	Qué controla
`enforcement.autoBan`	`enabled`, `detectors`, `categories`	Lista de ban instantáneo. De fábrica: solo `honeypot.trap`.
`enforcement.autoKick`	`enabled`, `minCount`, `detectors`	Lista de kick por señal sostenida (4).
`enforcement.appealUrl`	URL	Enlace de apelación de la tarjeta de baneado. Cámbialo.
`enforcement.defaultFailMode`	`open`, `closed`	Qué hace un chequeo no crítico cuando el propio Aegis falla: `open` (por defecto) deja pasar la acción; `closed` la deniega. Déjalo en `open` salvo que sepas por qué.
`enforcement.tempBanSeconds`, `enforcement.quarantine`	`blockRewards`, `limitEntities`, `blockEconomy`, `isolateBucket`, `alertStaff`	Duración del ban temporal; qué restringe la cuarentena (ver 15.1). `isolateBucket` (aislamiento en routing bucket) es opt-in y viene apagado.
`detectorActions`	`{ ['detector.id'] = 'off', 'log', 'kick', 'ban' }`	Override por detector, lo escribe la sección Detecciones.
`weaponPolicy`	`{ WEAPON_X = 'legal', 'warn', 'kick', 'ban' }`	Política por arma, la escribe Ajustes.
`audit`	`ringCapacity`, `keepSignalsPerSession`, `persist`, `persistMinSeverity`, `retentionDays`	Buffers en memoria, persistencia en base de datos y poda (30 días).
`entityguard`	`limits`, `burst`, `severity`, `signalWindowMs`, `forbiddenExplosions`, `explosion.ignoreTypes`, `spawnLimits`, `deleteOnMassSpawn`	Presupuestos de entidades y reglas de explosiones (7).
`economy`	`player`, `resource`, `breaker`, `severity`, `ignoreReasons`, `signalWindowMs`	Anomalías de ganancias y breakers (8).
`sensors`	`heartbeatMs`, `reportRate`, `challenge`, `movement`, `confidence`, `godmode`, `loadedGraceMs`	Entrada de sensores de cliente, invariantes de movimiento, opt-in anti-tamper.

Sección	Claves	Qué controla
`combat`	`snapDegPerSec`, `minSnapDeg`, `snapToShotMs`, `reactionMs`, `analyzeFrames`, `minShots`, `suspiciousRatio`, `damageModFactor`, `damageModMinDamage`, `damageModAbsolute`, `firerateIgnore`, `countNpcTargets`	Umbrales de Combat Intelligence (9). `minSnapDeg` (35°): un giro menor que esto nunca es un snap, así que los microajustes se ignoran. `snapToShotMs` (200 ms): el disparo tiene que llegar dentro de este margen tras el snap para contar. `analyzeFrames` (40): frames recientes de telemetría analizados por disparo. `damageModMinDamage` (100): `combat.damage_mod` solo salta si el daño sobrescrito es además al menos este valor absoluto. `damageModAbsolute` (1000): para armas sin entrada en el catálogo, solo marca un valor igual o superior a este.
`chat`	`max`, `window`, `repeatMax`, `severity`	Anti-spam de chat.
`quality`	`autoShadow`, `minSamples`, `fpRateDemote`, `evalWindow`	Degradación automática de detectores ruidosos.
`evidence`	`preMs`, `postMs`, `telemetryHz`, `bufferSeconds`, `maxCases`, `maxFramesPerCase`	Captura de casos de evidencia.
`review`	`maxQueue`	Tamaño de la cola de revisión.
`learning`, `honeypot`, `capabilities`		Ventana de aprendizaje de tráfico (funciona en modo Test), rotación del honeypot (sin semillas por defecto), TTL de los tickets de capacidad.
`screenshots`	`enabled`, `webhook`, `quality`, `burst`, `cooldownMs`, `minRisk`	Capturas de evidencia con `screenshot-basic`: ráfaga de 4, 30 s de cooldown por jugador, solo cuando el riesgo es al menos 80 o Aegis está ejecutando. `webhook` envía las imágenes a tu Discord en vez de guardarlas en el servidor.
`license`	`endpoint`, `recheckMinutes`, `bootGraceMinutes`	Informativo: el endpoint del portal viene del módulo cifrado. No hay key que configurar; el servidor se identifica con el código de portal (ver 2.1).

Sección	Claves	Qué controla
`trustedResources`	`{ 'nombre_recurso', ... }`	Recursos autorizados a llamar a los exports que quitan protección o inyectan señales (ver 15.1). Por defecto `{}`: solo el propio Aegis y el bootstrap del sidecar de los recursos protegidos.
`investigate`	`spectateAce`, `entityCapture`, `entityCaptureMax`	Permiso para espectar y captura en vivo de entidades.
`debug`	booleano	Log detallado.

12. Licencia y control remoto

- No hay key que configurar. El servidor se identifica ante el portal con un hash de un id aleatorio persistido en `data/serverid.txt` (FXServer oculta `sv_licenseKey` a los recursos, así que la licencia nunca se lee, y mucho menos se envía) y un código de portal corto derivado de ese hash. Vincular es opcional y se hace desde el portal; los servidores sin vincular protegen con normalidad.
- Aegis valida al arrancar y cada 30 minutos contra `https://meridianrp.es/aegis/api/validate`. La respuesta va firmada con HMAC; una respuesta sin firma o mal firmada se ignora.
- Una licencia vincula hasta 2 servidores (producción y desarrollo). Vincular y desvincular se hace en el portal con el código de portal; un tercer vínculo se rechaza hasta que desvincules uno, salvo que uno de los servidores vinculados lleve 14 días sin verse, en cuyo caso ese hueco se libera solo.
- El backend nunca apaga la protección. Si no responde, Aegis conserva el último estado firmado en una caché local. Si la licencia vinculada ha caducado, está suspendida o revocada, Aegis sigue protegiendo y avisa en consola para que lo revises en el portal. El acceso al recurso en sí lo gobiernan Tebex y el Asset Escrow.
- El proveedor puede, para toda la flota o para un servidor, forzar el modo Test, apagar el enforcement (kill switch) o enviar una revisión de ajustes. Todo va firmado y solo se usa ante un bug grave en una versión. La barra de estado del panel lo muestra.
- Los avisos de actualización también van firmados: una versión más nueva imprime una línea en consola con la URL de descarga; nada se descarga ni se ejecuta solo.

13. Privacidad

En claro, porque importa:

- Aegis **no lee ficheros** del PC del jugador. No tiene driver de kernel, ni escáner de ficheros, ni lista de procesos.
- Aegis **no tiene banlist global**. Los bans son tuyos, se guardan en tu servidor o en tu base de datos, y nunca se comparten con otros servidores ni con el proveedor.

- Aegis **no envía capturas ni datos de jugadores al proveedor**. Las capturas se quedan en tu servidor (ring en memoria y casos de evidencia) o van al webhook de Discord que **tú** configures en `screenshots.webhook`.
- Lo que recibe el backend de licencias cada 30 minutos: un hash SHA-256 del id de servidor aleatorio de `data/serverid.txt` (tu licencia de Cfx nunca se lee), el código de portal, la IP pública del servidor (como origen de la petición HTTPS), `sv_hostname`, la versión de Aegis y **contadores agregados** (jugadores conectados, señales, acciones, bans, modo). Sin identificadores, nombres, posiciones, chat ni imágenes.
- El paquete de soporte (`aegis_support` o Diagnóstico, Exportar) contiene configuración y chequeos del entorno sin secretos; tú decides si lo envías.

14. Problemas frecuentes y FAQ

La consola muestra "Servidor sin vincular al portal (opcional). Código de portal: XXXX-XXXX". Es informativo. Pega el código en https://meridianrp.es/aegis/ → "Vincular servidor" si quieres ver este servidor en tu portal y recibir avisos de versión. La protección no cambia. `aegis_portal` vuelve a mostrar el código.

El portal dice "código no encontrado" al vincular. El servidor aún no ha contactado con el backend (lo hace al arrancar y cada 30 minutos). Comprueba que Aegis está arrancado y que el servidor tiene salida HTTPS, espera un minuto y reintenta.

El portal dice "slots completos". Ya hay dos servidores vinculados y los dos se han visto en los últimos 14 días. Desvincula uno desde el portal y reintenta. Un servidor vinculado que lleve 14 días sin contactar con el backend se libera solo al vincular uno nuevo, así que un servidor de pruebas muerto nunca te bloquea mucho tiempo.

He borrado `data/` y el código de portal ha cambiado. Es lo esperado: el código sale del id aleatorio de `data/serverid.txt`, así que un `data/` nuevo significa un id nuevo. Vuelve a vincular el servidor desde el portal con el código nuevo; la entrada antigua se libera sola (al momento si la desvinculas, o a los 14 días sin contacto). La protección no se ve afectada.

El panel no abre o dice "sin permiso". El jugador necesita `aegis.admin` (revisa `add_ace group.admin aegis.admin allow` y que el jugador esté de verdad en `group.admin`) o estar en la lista "Quién puede usar /aegis". Desde consola, `aegis_status` te dice si el recurso está corriendo.

No hay capturas en la evidencia. Instala `screenshot-basic` y comprueba que la carpeta `.tmp` de dentro tiene permisos de escritura. Aegis avisa al arrancar si no puede crearla. Solo se piden capturas cuando el riesgo es 80 o más, o cuando Aegis está ejecutando, con 30 s de cooldown por jugador.

Los bans se pierden al reiniciar. Arranca `oxmysql` (las tablas se crean solas) o asegúrate de que `data/` dentro del recurso es escribible para que se pueda escribir `data/bans.json`. La consola imprime "bans en data/bans.json" o "persistencia lista" al arrancar.

Se bloquea un evento legítimo. Abre Escáner de recursos, el recurso, el evento, y relaja o elimina el contrato (rate demasiado bajo, distancia demasiado corta, sesión que falta). O Desprotege el recurso. Recuerda que en Test no se bloquea nada, así que si se bloquea estás en Activo.

Falsos positivos. Deja Aegis en Test 1 o 2 días, revisa Incidentes, pon el detector ruidoso en Registrar en Detecciones, declara tus spawns con `EntityExpect`, registra las entregas de armas con `GrantWeapon`, y solo entonces pasa a Activo. Marca los falsos positivos en Revisión o con `aegis_fp` para que se retire el riesgo y el detector aprenda.

Aegis imprime "CONFIG RECHAZADA" y no arranca. `shared/config.lua` contiene código que no es una tabla de configuración. Deshaz el cambio o restaura el fichero desde el zip.

El modo vuelve a Test tras reiniciar. Pulsa "Guardar cambios" en el panel después de cambiar el modo, o fija `mode` en `config.lua`.

Los mensajes salen en otro idioma. Los mensajes de desconexión, la tarjeta de baneado y el anuncio público siguen `locale`. Cámbialo con `aegis_lang` o en Ajustes.

15. Exports (API de servidor)

Todos los exports son de servidor y se llaman como `exports.meridian_aegis:Nombre(...)`.

Núcleo:

- `ReportSignal(source, detector, data)`: reporta evidencia para un detector registrado; severidad y confianza vienen del detector, no de quien llama.
- `RegisterDetector(def)`: registra tu propio detector (`id`, `category`, `severity`, `confidence`, `description`).
- `GetRisk(source)`: riesgo actual de un jugador.
- `GetSession(source)`: instantánea de la sesión de seguridad del jugador.
- `IsHealthy()`: true si el runtime está sano.
- `GetLocale()`: idioma actual de Aegis.
- `Quarantine(source, reason)` y `Release(source)`: meter o sacar a un jugador de cuarentena.

EventGuard:

- `ProtectEvent(name, policy)`: declara un contrato para un evento.
- `RegisterProtectedEvent(name, policy, handler)`: declara el contrato y registra el handler, validado antes de ejecutarse.
- `CheckEvent(source, event, ctx)`: validación manual; devuelve `{ allow, sanitized, ... }`.
- `IssueNonce(source, op, ttl)`: nonce de un solo uso para anti-replay.
- `StartSecuritySession(source, kind, context, ttl)`: abre una sesión y devuelve su id.
- `SetSessionStage(id, stage)` y `EndSecuritySession(id)`: avanzan o cierran una sesión.

ResourceShield e instalador:

- `ScanResource(name)`, `ScanAll()`, `PlanResource(name)`: escaneo de solo lectura y propuestas de contratos.
- `ProtectResource(res)` y `UnprotectResource(res)`: el flujo completo de Proteger/Desproteger que usa el panel.
- `SetContract(res, event, policy)` y `RemoveContract(res, event)`: gestionan contratos del sidecar.
- `PlanInstall(res)`, `ApplyInstall(res, opts)`, `RollbackInstall(res)`, `Unprotect(res)`: integración del manifest con copia de seguridad y rollback.
- `SidecarHello(res)`, `SidecarContracts(res)`, `SidecarCheck(res, source, event, packed)`: los usa el propio sidecar; no los llamas tú.

EntityGuard, economía, armas y movimiento:

- `EntityExpect(resource, kind, count, ttl)`: declara spawns legítimos inminentes.
- `GuardReward(ctx)`: comprueba una recompensa contra breakers y anomalías; devuelve `{ allow }`.
- `OnEconomyDelta(kind, ctx)`: alimenta un delta económico propio.
- `ResetBreaker(res, op)`: rearma un circuit breaker.
- `GrantWeapon(source, weapon, ttlMs)`: registra una entrega legítima de arma.
- `AllowTeleport(source, ttl)`: exime un teleport legítimo (herramientas de staff, scripts).

Capacidades, aprendizaje e informes:

- `GrantCapability(source, capability, scope)`, `HasCapability(source, capability)`, `ConsumeCapability(source, capability)`, `RequestCapability(source, capability, scope)`, `CapabilityCatalog()`: tickets temporales (`teleport`, `noclip`, `invisibility`, `spectate`, `spawn_vehicle`, etc.) para que una acción legítima no se marque.
- `ProposeContract(event)`: propuesta de contrato para un evento, construida con el tráfico observado en modo Test.
- `SecurityReport()`: el Aegis Security Report como tabla.
- `SupportBundle()`: el paquete de soporte como tabla (sin secretos).

15.1 Recursos de confianza y qué hace de verdad la cuarentena

Un cheater que consiga un punto de apoyo en el servidor (un recurso comprometido, un ejecutor con módulo de servidor) no debe poder convencer a Aegis de que se apague solo. Por eso los exports que quitan protección o inyectan señales solo funcionan cuando el recurso que llama está en `Config.trustedResources` (una tabla de nombres de recurso, por defecto `{}`, que significa solo el propio Aegis y el bootstrap del sidecar de los recursos que protege). Los exports vigilados son `ReportSignal` cuando el detector es hard, `Unprotect`, `UnprotectResource`, `Release`, `AllowTeleport`, `GrantCapability` y `SidecarHello`. Una llamada desde cualquier otro recurso se rechaza, no hace nada y queda registrada en consola y en la auditoría de staff. Los exports de solo lectura (`GetRisk`, `ScanResource`, `SecurityReport`, etc.) y los que añaden protección (`ProtectEvent`, `EntityExpect`, `GrantWeapon`, `GuardReward`) siguen abiertos a todos los recursos.

La cuarentena es una restricción real, no solo una marca. Mientras un jugador está en cuarentena, y según `enforcement.quarantine`: las recompensas que pasan por `GuardReward` se deniegan (`blockRewards`), el presupuesto de entidades se reduce para cortar los mass spawns (`limitEntities`), y los hooks de dinero del framework y los eventos protegidos de ese jugador se deniegan (`blockEconomy`). Se avisa al staff (`alertStaff`) y el jugador se suelta desde la ficha o con `Release`.

16. Soporte

- Portal del comprador, descargas y estado de la licencia: https://meridianrp.es/aegis/
- Documentación online: https://meridianrp.es/aegis/docs (español) y https://meridianrp.es/aegis/docs?lang=en (inglés); PDFs en https://meridianrp.es/aegis/docs/DOCUMENTATION.pdf y https://meridianrp.es/aegis/docs/DOCUMENTATION.es.pdf
- Soporte por tickets de Discord: https://discord.gg/YkCtxAvbPD, botón "Abrir ticket" en #support (también avisos de versión y reportes de falsos positivos).
- Antes de abrir un ticket, ejecuta `aegis_diagnostics` y `aegis_support`, y adjunta `data/aegis-support.json`. No contiene secretos.
- Incluye las líneas de consola de tu servidor que empiezan por `[Aegis` alrededor del problema.