SUITE SHADOW

ShadowQuests

Missioni infinite, premi di qualunque tipo, e un editor che si usa in gioco. La potenza di un plugin per programmatori, con una finestra al posto del linguaggio.

VERSIONE	MINECRAFT	SERVER	JAVA
1.0.0	1.26.2	Paper 26.2	25

Manuale d'uso · Funzionalità, comandi, permessi, dipendenze e integrazioni, configurazione, domande frequenti

Indice

PERCHÉ NEL MANUALE SI LEGGE `/QUEST`

Il comando si chiama `/shadowquests` e ha gli alias `/quest` e `/sq`. Un alias però non è garantito: se un altro plugin ha già preso `/sq`, chi arriva dopo non lo ottiene. All'avvio la console scrive quale etichetta ha ottenuto davvero e quali sono andate perse, e ogni messaggio del plugin nomina quella vera. Qui sotto scriviamo `/quest` perché si legge meglio.

Come leggere questo manuale

Se vuoi solo partire, ti bastano i capitoli **2** e **5**: installi, apri l'editor e scrivi la prima missione in cinque minuti. Il capitolo **6** serve solo a chi preferisce i file. Tutto il resto serve quando vorrai sapere perché una cosa funziona come funziona.

1. Che cos'è ShadowQuests

ShadowQuests fa missioni. Quante ne vuoi, con qualunque premio, e si costruiscono da una finestra in gioco invece che scrivendo un linguaggio dentro a un file di testo.

Il problema

I plugin di missioni che esistono si dividono in due gruppi, e nessuno dei due è quello che serve alla maggior parte dei server.

Da una parte c'è chi è potentissimo e si configura scrivendo: un linguaggio proprio, dentro a file di testo, con eventi, condizioni e variabili. Chi sa programmare ci costruisce qualunque cosa. Chi non sa programmare si arrende alla seconda missione, e quel «chi non sa programmare» è la maggioranza delle persone che aprono un server.

Dall'altra c'è chi è accessibile e vecchio: una GUI che fa poco, tre tipi di obiettivo, e il momento in cui vorresti una missione un po' particolare è il momento in cui scopri che non si può fare.

La risposta

Cosa fa	Come
Missioni senza limite	Nessun tetto al numero, ai gruppi, agli obiettivi per missione o ai premi.
Settanta tipi di obiettivo	Blocchi, creature, item, movimento, il giocatore, l'economia, il resto della suite. E l'elenco si allunga senza toccare il motore.
Premi di qualunque tipo	Item (anche di altri plugin), soldi, token, comandi, permessi a tempo, chiavi delle crate, punti del battle pass, premi a sorte e a scelta.
Un editor in gioco	Una missione intera - nome, icona, obiettivi, filtri, premi, catena - senza aprire un file. Con l'anteprima di come la vedrà un giocatore.
E anche gli YAML	Chi preferisce scrivere continua a scrivere. L'editor produce gli stessi file, con le stesse chiavi: non esiste un formato «dell'editor».
Cicli veri	Giornaliere, settimanali, mensili, annuali, con l'ora e il fuso che scegli tu. E che funzionano anche se il server era spento all'ora prevista.

La missione che nessun concorrente può scrivere

«Vinci 3 partite a BedWars, guadagna 500 monete e apri una crate leggendaria». Sono tre obiettivi che vivono dentro a tre plugin diversi - ShadowGames, ShadowEconomy, ShadowCrazyChest - e nessun plugin di missioni può vederli se non possiede quei tre plugin. ShadowQuests li vede, perché fa parte della stessa suite.

Nessuna delle integrazioni è obbligatoria: quello che manca spegne una riga e non l'avvio, e la cosa viene scritta in console una volta sola con il nome di cosa manca.

PESO SUL SERVER: QUELLO CHE SERVE, E NIENT'ALTRO

ShadowQuests registra **solo gli ascoltatori che almeno una missione usa davvero**. Su un server con dieci missioni sui blocchi, l'ascoltatore delle entità, quello degli item e quello del movimento non sono «leggeri»: non esistono. Quando cancelli l'ultima missione che usava un tipo, il suo ascoltatore viene tolto - subito, senza riavviare.

L'unico ascoltatore sul movimento - quello delle missioni da camminare - filtra sul blocco cambiato prima di fare qualunque cosa, e comunque esiste solo se hai scritto una missione da camminare.

2. Requisiti e installazione

Requisiti

Minecraft	1.26.2
Server	Paper 26.2 (build 115 o successiva). Compatibile con Folia.
Java	25, che è comunque il requisito di Paper 26.2
Dipendenze	Nessuna. Nessun database da installare, nessuna libreria da mettere sul server.
Spazio	Circa 4 MB per il jar: dentro ci sono il pool di connessioni e i driver, così non devi installarli tu.

Installazione

1. **Copia il jar.** Metti `ShadowQuests-1.0.0.jar` nella cartella `plugins` del server.
2. **Avvia il server.** Al primo avvio vengono creati la configurazione, le lingue, i menu e tre file di missioni d'esempio, che sono già giocabili.
3. **Entra in gioco e scrivi `/quest`.** Le missioni ci sono già.
4. **Scrivi la tua prima missione** con `/sqa editor`. Vedi il capitolo 5.

Alla fine, nella cartella dati troverai questo:

```
plugins/ShadowQuests/
  config.yml                tutte le impostazioni
  license.yml               la licenza, tenuta separata dal config
  rewards.yml               i pacchetti di premi riutilizzabili
  npcs.yml                  quale NPC consegna quale missione
  quests/daily.yml          un gruppo di missioni giornaliere
  quests/miner-chain.yml    una storia in quattro capitoli
  quests/examples.yml       due missioni scritte per essere lette
  lang/it_IT.yml            i messaggi in italiano
  lang/en_US.yml            i messaggi in inglese
  menus/*.yml               la disposizione delle finestre
  data/quests.mv.db         il database locale, creato da solo
  data/installed-files.txt  contabilita' del plugin, non si tocca
```

CHE LINGUA PARLANO I FILE

Tutto quello che si *configura* e' scritto in inglese: nomi delle proprieta', sezioni, valori e i commenti che li spiegano. Il plugin si vende in tutto il mondo, e un file mezzo in una lingua e mezzo in un'altra e' un file in cui non si capisce quale delle due il plugin ascolti davvero. Lo stesso vale per i comandi: si scrivono solo in inglese.

Quello che *legge un giocatore* e' un'altra cosa e resta traducibile: sta nei file `lang/`, arriva in italiano e in inglese, e la lingua predefinita si sceglie con `messages.default-locale`. Anche i nomi e le descrizioni degli oggetti dentro ai menu sono testo da leggere, e infatti arrivano in italiano.

FUNZIONA SUBITO, SENZA TOCCARE NIENTE

Il database non va installato: senza configurare niente il plugin usa un file locale dentro alla propria cartella. MySQL serve solo se hai una rete di server che devono condividere le stesse missioni, ed è una funzione Premium.

Aggiornare

Si sostituisce il jar e si riavvia. `config.yml` e i file dentro a `quests/` non vengono **mai** sovrascritti: le sezioni nuove di una versione futura vengono accodate in fondo al tuo config, con i loro commenti, senza toccare una riga di quello che hai scritto tu.

I file dentro a `lang/` e `menus/` invece vengono aggiornati, perché lì ci sono le parole e la disposizione e lasciare la copia vecchia vorrebbe dire che un aggiornamento non cambia mai niente di quello che si legge. La tua copia viene salvata accanto come `.old`, quindi non perdi niente.

SE VUOI CAMBIARE I TESTI IN MODO PERMANENTE

Non modificare `lang/it_IT.yml`: fai una copia con un nome tuo, per esempio `lang/it_CUSTOM.yml`, cambia `meta.locale` dentro al file e imposta quel nome come `messages.default-locale` in `config.yml`. Così nessun aggiornamento lo tocca.

3. Gratuito e Premium

Un jar solo, due versioni. Senza chiave gira la **gratuita**, che è un prodotto intero e non una dimostrazione. Con una chiave firmata gira la **Premium**, che toglie i tre limiti che contano quando un server comincia ad avere gente.

Cosa cambia

	Gratuita	Premium
Missioni caricate	**10**	senza limite
Cicli automatici (giornaliere, settimanali, mensili, annuali)	no	sì
MySQL e più server che condividono le missioni	no	sì
Tipi di obiettivo	tutti	tutti
Tipi di ricompensa	tutti	tutti
Editor in gioco	sì	sì
Catene, prerequisiti, missioni ripetibili	sì	sì
GUI personalizzabili, due lingue, API	sì	sì
Integrazioni con il resto della suite	tutte	tutte
Una riga di credito dopo un premio	sì, una per sessione	no

COME SI COMPORTA UN LIMITE

Si applica in entrata, mai togliendo qualcosa a metà. Una missione già cominciata si finisce.

Niente di costruito si perde. L'undicesima missione non viene caricata, ma resta su disco intatta e torna da sola appena arriva la chiave. Nessun file viene toccato, nessun progresso viene cancellato.

Le funzioni Premium non si registrano affatto, così il resto non si rompe: un gruppo dichiarato giornaliero su un server gratuito semplicemente non si rinnova, e le sue missioni continuano a funzionare.

Attivare una licenza

1. **Compra.** Ricevi un codice della forma `SQ-XXXX-XXXX`.
2. **Scrivi `/quest license claim <codice>`.** Il server chiede da solo la licenza al servizio di attivazione, che gliela firma legata ai suoi indirizzi. Un secondo, nessuna persona coinvolta.
3. **Fatto.** La chiave viene scritta in `license.yml` e il plugin passa a Premium senza riavviare.

Se hai già una chiave in mano, `/quest license activate <chiave>`. Per vedere lo stato in qualunque momento, `/quest license`.

IL PLUGIN NON SI SPEGNE MAI PER UN PROBLEMA DI LICENZA

Un abbonamento appena scaduto o un indirizzo che non corrisponde danno **sette giorni di tolleranza** - il secondo caso è molto più spesso un NAT o un proxy che un ladro. Passati quelli, il plugin torna alla versione gratuita e lo scrive chiaramente in console. Non si disabilita, non lancia errori, non interrompe una missione a metà.

LA CHIAVE NON SI VEDE MAI PER INTERO

Nemmeno a un operatore, nemmeno in console. `/quest license` la mostra mascherata, perché lo screenshot di quel comando finisce prima o poi in un canale di assistenza.

4. Funzionalità

Una missione

Una missione è un nome, un'immagine, un elenco di cose da fare e un elenco di cose da ricevere. Tutto il resto è facoltativo: la missione utile più corta è sei righe, e quella più elaborata ha la stessa forma.

Gli obiettivi

Una missione può averne quanti ne vuoi, e decide se servono **tutti** oppure **uno qualunque**. Le categorie sono queste:

Categoria	Qualche esempio
Blocchi	rompere, posizionare, scavare minerali, raccogliere, tagliare legna, riempire secchi, aprire contenitori, rompere spawner
Creature	uccidere mob, giocatori, boss, mob di altri plugin, accoppiare, addomesticare, tosare, curare villici, vincere incursioni, danno inflitto e subito
Item	fabbricare, cucinare, incantare, mangiare, raccogliere, buttare, distillare, pescare, riparare, commerciare, incudine, mola, tavolo da fabbro
Movimento	camminare, correre, nuotare, volare, navigare, cavalcare, visitare biomi e mondi, raggiungere un punto preciso
Giocatore	tempo di gioco, accessi, morti, notti dormite, chat, comandi, progressi, portali, livello di esperienza, salti, voti
Economia	guadagnare, spendere, comprare e vendere al negozio, aste, banca, token, criptovalute, valore del portafoglio
Suite	aprire crate, vincere e giocare partite, uccisioni in partita, livello dell'isola, livello del battle pass, parlare con un NPC

I filtri

È quello che trasforma settanta tipi in migliaia di missioni diverse. Ogni obiettivo può restringere:

- **Il bersaglio** - quali materiali, quali creature, quale nome.
- **Il mondo, il bioma, la regione** - «scava nella miniera» è una missione diversa da «scava».
- **La quota** - da -64 a 16 significa «in profondità».
- **Come** - solo con il Tocco di Velluto, solo mob ostili, solo mob non nati da uno spawner.
- **L'item** - il suo nome, la sua descrizione, i suoi incantesimi.
- **Chi** - qualunque requisito: permesso, livello, soldi, missioni già finite, data, probabilità.

UN FILTRO CHE NON PUÒ CORRISPONDERE TE LO DICE

Se scrivi un materiale su un obiettivo di uccisioni, il plugin lo scrive in console al caricamento: *«DIAMOND non è un tipo di entità, quindi mob_kill non lo conterà mai»*. È il guasto più segnalato di tutti i plugin di questo genere - una missione che non si completa mai e nessuno sa perché - e qui si vede il giorno in cui la scrivi.

Le ricompense

Tipo	Cosa fa
`ITEM`	Un item vero, anche di Oraxen, Nexo, ItemsAdder, MMOItems o ShadowCustomizer. Le quantità grandi vengono spezzate in pile.
`MONEY`	Soldi, da ShadowEconomy o da Vault.
`TOKEN`	La seconda valuta di ShadowEconomy.
`PASS_XP` / `PASS_TIER`	Punti e livelli del battle pass della suite.
`VANILLA_XP`	Esperienza di Minecraft, in punti o in livelli.
`COMMAND`	Uno o più comandi dalla console, con `<player>` sostituito.
`PERMISSION`	Un permesso, anche a tempo, tramite ShadowPermissions o LuckPerms.
`CRATE_KEY`	Chiavi di una crate, con il comando del tuo plugin di crate.
`SCRIPT`	Righe nel linguaggio di ShadowCommand.
`RANDOM`	Una fra tante, estratta a peso.
`CHOICE`	Una fra tante, scelta dal giocatore in una finestrella.
`BUNDLE`	Un pacchetto definito una volta in `rewards.yml` e richiamato da ovunque.
e poi	messaggi, annunci a tutto il server, titoli, suoni, fuochi d'artificio, effetti.

OGNI PREMIO FINISCE IN UN REGISTRO

Chi ha ricevuto cosa, quando e per quale missione. Serve il giorno in cui qualcuno dice di non aver mai ricevuto un premio - e quel giorno arriva: una ricompensa che è un comando non lascia nessun'altra traccia, la console esegue e da lì in poi non c'è più niente da cui capire se è successo.

Le catene

Una storia in capitoli: ogni missione ne sblocca la successiva, e i capitoli non ancora raggiunti non si vedono. Si costruisce con tre righe - `hidden`, `requires` e `unlocks` - e si guarda con `/quest chains`, che mostra a che punto è ogni storia e qual è il prossimo capitolo.

I cicli

Un gruppo di missioni ha un ritmo: giornaliero, settimanale, mensile, annuale o nessuno. Il gruppo decide anche **quante missioni tenere per volta**: scrivine trenta, consegnane tre a caso ogni giorno, e hai tre mesi di contenuti senza scrivere tre mesi di contenuti.

> **LE GIORNALIERE SI REIMPOSTANO ANCHE SE IL SERVER ERA SPENTO**
>
> Non c'è nessun timer da far scattare a mezzanotte. Ogni missione è salvata sotto al *nome* del ciclo a cui appartiene - `2026-03-01`, `2026-W10` - e quando un giocatore entra si calcola il nome di adesso e lo si confronta con quello salvato. Se sono diversi, il ciclo è cambiato: che sia successo un minuto fa o durante tre giorni di server spento.
>
> L'ora della reimpostazione e il fuso orario si scelgono in `config.yml`. Il predefinito è le 04:00, e non è un caso: chi gioca all'una di notte sta ancora giocando la sera prima.

Le regole anti-abuso

Non sono un anticheat: sono tre regole di buon senso che tolgono di mezzo i modi più ovvi di far avanzare una missione senza giocarla.

- **Un blocco posato e poi rotto non conta.** Senza questa, «scava 500 minerali» si risolve mettendo e rompendo lo stesso blocco.
- **I mob nati da uno spawner non contano** per le missioni sulle uccisioni. È l'unica difesa vera contro le trappole industriali.
- **Chi è fermo da un po' non fa avanzare niente**, e c'è un tetto agli eventi che possono contare in un secondo.

Tutte e tre si spengono in `config.yml`, e ogni singola missione può decidere diversamente per conto suo.

5. L’editor in gioco

`/sqa editor`. Da lì si costruisce una missione intera senza aprire un file, e si guarda com’è venuta con la stessa finestra che vedrà un giocatore.

Come si fa una missione

1. **Apri l’editor** con `/sqa editor` e premi *Nuova missione*. Ti viene chiesto un nome breve: sarà il suo id.
2. **Riempi le caselle.** Nome, descrizione, icona, gruppo. Le cose corte si scrivono in chat e la finestra si riapre da sola.
3. **Aggiungi gli obiettivi.** Si scelgono da un elenco diviso per categoria, e subito dopo ti viene chiesto quante volte e su cosa.
4. **Aggiungi le ricompense.** Il modo più veloce è «*aggiungi l’item che hai in mano*»: prende quello che stai tenendo, con nome, incantesimi e tutto il resto.
5. **Guarda l’anteprima.** Si apre la finestra vera, non una schermata di prova: quello che vedi è quello che vedrà chi gioca.
6. **Salva.** La missione finisce in un file dentro a `quests/`, il plugin ricarica da solo e la missione è viva.

> **L’ICONA SI PRENDE DALLA MANO**
>
> Quando l’editor chiede l’icona, rispondi `mano` e prende l’item che stai tenendo. Lo stesso vale per le ricompense.

> **IL PUNTO DA RAGGIUNGERE SI PRENDE DAI PIEDI**
>
> Per una missione «arriva qui» non si scrivono tre coordinate: si va sul posto, si preme il pulsante, e l’editor usa il punto in cui sei. Poi chiede il raggio.

Cosa fa l’editor e cosa no

L’editor copre tutto quello che serve a scrivere una missione: obiettivi con i loro filtri, ricompense, requisiti, catene, tempo massimo, quante volte si ripete, quale NPC la consegna.

Le due cose che **non** fa sono i premi `RANDOM` e `CHOICE`, perché hanno bisogno di ricompense annidate dentro ad altre ricompense e un editor dentro a un editor sarebbe più difficile da usare che scriverlo. Si scrivono in `rewards.yml` - ci sono già due esempi pronti - e poi si richiamano dall’editor come pacchetto.

> **L’EDITOR E I FILE SONO LA STESSA COSA**
>
> Quello che l’editor scrive è un normale file YAML con le normali chiavi: si può aprire con un editor di testo, capire, e modificare a mano. E una missione scritta a mano si può aprire nell’editor. Non esistono due formati.
>
> La bozza vive in memoria finché non premi salva: chi esce dal gioco a metà non lascia in giro mezza missione.

6. Scrivere le missioni a mano

Un file dentro a `quests/` è un gruppo. Il gruppo decide il ritmo, le missioni stanno sotto.

La forma più corta che funziona

```
quests:
  mine_stone:
    display: "<white>Il minatore di tutti i giorni"
    objectives:
      - type: block_break
        target: [ STONE ]
        amount: 300
    rewards:
      - "money: 500"
```

Senza blocco `group:` il file è un gruppo che non si reimposta mai, tiene tutte le sue missioni e si chiama come il file.

Il gruppo

```
group:
  id: giornaliere
  display: "<aqua>Giornaliere"
  icon: CLOCK
  order: 1
  period: DAILY          # DAILY | WEEKLY | MONTHLY | YEARLY | NEVER
  size: 3                # quante se ne tengono per volta, o "all"
  sequential: false      # true = in ordine, per le storie
  completion-rewards:
    - "money: 2500"
```

UN PERIODO SCRITTO MALE VIENE DETTO IN CONSOLE

Se scrivi `period: settimanle`, il plugin lo segnala all'avvio e usa `NEVER`. Senza quell'avviso ti troveresti delle settimanali che non si reimpostano mai e nessun modo di capire il perché.

Un obiettivo con tutti i filtri

```
objectives:
  - type: ore_mine
    target: [ DIAMOND_ORE, DEEPSLATE_DIAMOND_ORE ]
    amount: 10
    world: [ world ]
    biome: [ desert, badlands ]
    region: [ miniera ]
    min-y: -64
    max-y: 16
    silk-touch: true
    hostile-only: false
    no-spawner-mobs: true
    description: "<gray>Scava diamanti nelle profondita"
    conditions:
      - "permission: rank.miner"
```

Le ricompense, in breve

```
rewards:
  - "money: 5000"
  - "item: DIAMOND 3"
  - "item: oraxen:mithril_sword"
  - "token: gemme 10"
  - "pass-xp: 250"
  - "crate: leggendaria 1"
  - "permission: grado.vip 30d"
  - "command: give <player> golden_apple 3"
  - "script: message: &aBravo!"
  - "broadcast: <gold><player> ce l'ha fatta!"
  - "bundle: big-reward"
```

I requisiti

Un linguaggio solo, imparato una volta, uguale ovunque compaia la parola `conditions`:

```yaml
conditions:
  - "permission: quests.vip"
  - "world: world, world_nether"
  - "not-world: lobby"
  - "level: >= 20"
  - "money: >= 5000"
  - "quest-done: miner_1"
  - "quests-completed: >= 10"
  - "island-level: >= 100"
  - "y: 0..64"
  - "biome: desert"
  - "region: miniera"
  - "time: 13000..23000"
  - "weather: rain"
  - "date: 2026-12-01 .. 2026-12-31"
  - "chance: 25%"
  - "online: >= 10"
  - "placeholder: %player_level% >= 30"
  - "scmd: <una riga nel linguaggio di ShadowCommand>"
```

PERCHÉ C'È SCMD:

Perché ShadowCommand ha già un linguaggio per i requisiti, con le sue variabili e il suo editor, e inventarne un secondo qui vorrebbe dire chiedere alla stessa persona di impararne due. Quello che sta dopo `scmd:` viene passato pari pari alla sua API.

Quando ShadowCommand non c'è, quella condizione vale **vero**. La direzione della rinuncia è scelta: un'integrazione che manca deve rendere una missione *più facile*, mai impossibile da finire senza che nessuno capisca perché.

Una catena

```
group:
  id: minatore
  period: NEVER
  size: all
  sequential: true

quests:
  miner_1:
    display: "Capitolo I"
    chain: "La via del minatore"
    step: 1
    objectives: [ { type: block_break, target: [ STONE ], amount: 64 } ]
    rewards: [ "money: 500" ]
    unlocks: [ miner_2 ]

  miner_2:
    display: "Capitolo II"
    hidden: true                # non si vede finche' non tocca a lui
    requires: [ miner_1 ]
    chain: "La via del minatore"
    step: 2
    objectives: [ { type: ore_mine, target: [ IRON_ORE ], amount: 64 } ]
    rewards: [ "money: 2500" ]
```

PRIMA DI DIRE CHE È A POSTO

`/sqa check` cerca le missioni che non si possono finire: prerequisiti che non esistono, missioni nascoste senza niente che le sblocchi, gruppi vuoti, premi mancanti, NPC nominati su un server senza NPC.

7. Comandi

> **IN GIOCO NON SERVE SCRIVERE NIENTE**
>
> ShadowQuests si usa con i pulsanti: `/quest` apre la finestra e da lì è tutto a clic, comprese le conferme, che arrivano in chat come pulsanti verdi e rossi da premere. I comandi qui sotto esistono per la console e per chi li preferisce, non perché siano necessari.

Per i giocatori

Permesso: `shadowquests.use` (predefinito: tutti).

Comando	Cosa fa
`/quest`	Apre la finestra delle missioni.
`/quest list`	Le missioni in corso, in chat.
`/quest info <id>`	Una missione nei dettagli.
`/quest start <id>`	Accetta una missione che non parte da sola.
`/quest abandon <id>`	Lascia perdere una missione. Il progresso resta dov'è.
`/quest claim [id]`	Ritira un premio, o tutti quelli pronti.
`/quest chains`	Le catene.
`/quest lang [lingua]`	Sceglie la lingua, o `auto`.
`/quest help`	L'elenco in chat.

La licenza

Permesso: `shadowquests.admin.license` (predefinito: op).

Comando	Cosa fa
`/quest license`	Versione, limiti, licenza e impronta del server.
`/quest license claim <codice>`	Attiva con il codice d'acquisto. È il modo normale.
`/quest license activate <chiave>`	Attiva una chiave che hai già.
`/quest license refresh`	Rilegge la chiave da disco.

Per lo staff

Permesso: `shadowquests.admin` (predefinito: op).

Comando	Cosa fa
`/sqa editor`	L'editor in gioco.
`/sqa reload`	Rilegge configurazione, lingue, menu e missioni.
`/sqa info`	Stato: archivio, cicli, ascoltatori accesi, integrazioni, licenza.
`/sqa check`	Cerca le missioni che non si possono finire.
`/sqa quests`	L'elenco delle missioni caricate.
`/sqa types`	I tipi di obiettivo disponibili.
`/sqa give <giocatore> <id>`	Assegna una missione.
`/sqa complete <giocatore> <id>`	La completa, ricompense e catene comprese.
`/sqa progress <giocatore> <tipo> <n> [nome]`	Fa avanzare un tipo di obiettivo. Utile per provare.
`/sqa reset <giocatore>`	Azzera tutte le sue missioni. Chiede conferma.
`/sqa resetcycle <gruppo>`	Rimescola un gruppo per tutti. Chiede conferma.
`/sqa npc`	Chi consegna cosa. Con `bind` e `unbind` si cambia.

8. Permessi

Permesso	Predefinito	Cosa apre
`shadowquests.use`	tutti	Aprire le missioni e usare `/quest`.
`shadowquests.admin`	op	Tutti i comandi dello staff e l'editor in gioco.
`shadowquests.admin.license`	op	Vedere e attivare la licenza. Separato da `admin` apposta: chi amministra le missioni non è per forza chi amministra gli acquisti, e quel comando mostra l'impronta del server.
`shadowquests.bypass.limits`	op	Nessun controllo anti-AFK e nessun tetto di eventi al secondo. Da dare a chi prova le missioni, non ai giocatori.

I PERMESSI DENTRO ALLE MISSIONI

Quelli scritti nelle condizioni - `permission: grado.vip` - non appartengono a questo plugin e non vanno dichiarati da nessuna parte: li gestisce il tuo plugin dei permessi. Lo stesso vale per i permessi dati come ricompensa.

9. Dipendenze e integrazioni

Nessuna dipendenza obbligatoria. ShadowQuests si avvia e funziona su un server dove non c'è nient'altro. Ogni integrazione è facoltativa e sta dietro a un controllo di presenza.

Con il resto della suite

Plugin	Cosa aggiunge
ShadowEconomy	Premi in soldi e token; obiettivi su soldi guadagnati e spesi, negozio, aste, banca, criptovalute, portafoglio.
ShadowBattlePass	Premi in punti e livelli del pass; le missioni del pass fanno avanzare le nostre. **Gli obiettivi hanno gli stessi nomi nei due plugin**: quello che impari qui vale là, e si copiano da un file all'altro senza tradurre niente.
ShadowCrazyChest	Premi in chiavi; obiettivi sulle crate aperte.
ShadowGames	Obiettivi su partite vinte, giocate e uccisioni.
ShadowSkyBlock e le altre modalità	Obiettivi sul livello dell'isola.
ShadowCommand	Il linguaggio dei requisiti con `scmd:`, e le ricompense di tipo `SCRIPT`.
ShadowRegions	Il filtro `region:` e l'obiettivo «entra in una regione».
ShadowPermissions	Premi in permessi, anche a tempo.
ShadowCustomizer	Item e creature personalizzati, come premi e come bersagli.
ShadowBasics	Gli NPC: un personaggio può consegnare una missione e offrirla con due pulsanti.
ShadowControl	Il pannello sa che ci siamo e con che versione. Non decide niente sulla nostra licenza: quella la verifichiamo noi.

Fuori dalla suite

Plugin	Cosa aggiunge
Vault	L'economia, quando non c'è ShadowEconomy.
LuckPerms	I permessi come premio, anche temporanei.
WorldGuard	Il filtro sulle regioni, quando non c'è ShadowRegions.
PlaceholderAPI	I nostri segnaposto verso l'esterno e i loro dentro alle nostre condizioni.
Oraxen, Nexo, ItemsAdder, MMOItems	Item personalizzati come premio.
MythicMobs	Obiettivi sulle creature personalizzate.

Plugin	Cosa aggiunge
CrazyCrates ed equivalenti	Chiavi come premio, quando non c'è ShadowCrazyChest.
Votifier	L'obiettivo sui voti.

I segnaposto di PlaceholderAPI

```
%shadowquests_active%          quante missioni ha in corso
%shadowquests_completed%       quante ne ha finite adesso
%shadowquests_claimable%       quanti premi ha da ritirare
%shadowquests_completed_ever%  quante ne ha finite in tutto
%shadowquests_total%           quante missioni esistono
%shadowquests_reset_daily%     quanto manca alla prossima giornaliera
%shadowquests_reset_weekly%    e alla settimanale
%shadowquests_reset_monthly%   e alla mensile
%shadowquests_progress_<id>%   la percentuale di una missione
%shadowquests_done_<id>%       si / no per una missione
```

PER CHI SCRIVE PLUGIN

C'è un'API nel ServicesManager di Bukkit (`net.shadowmc.quests.api.ShadowQuestsAPI`) scritta con tipi semplici, così si può usare per riflessione senza compilare contro ShadowQuests. Permette di far avanzare obiettivi dall'esterno e - la parte interessante - di **registrare tipi di obiettivo nuovi**, che da quel momento compaiono nell'editor e nei file come tutti gli altri.

10. Configurazione

`config.yml` è commentato riga per riga e spiega il *perché* di ogni scelta, non solo il cosa. Qui c'è la mappa.

Archivio

Voce	Cosa fa
`storage.mysql.*`	Le credenziali. Con `enabled: false` si usa un file locale, che non va installato. MySQL è Premium.
`storage.table-prefix`	Il prefisso delle tabelle. Cambiandolo si ricomincia da zero: utile per due server sullo stesso database.
`storage.fallback-to-local`	MySQL irraggiungibile: con `true` il server parte lo stesso sul file locale.
`storage.flush-interval`	Ogni quanto il progresso accumulato finisce su database. All'uscita di un giocatore e allo spegnimento si scrive comunque.
`storage.history.*`	Il registro di chi ha ricevuto cosa, e per quanti giorni tenerlo.
`storage.sync.*`	Più server sullo stesso database. Premium. Su una rete, **un solo server** deve avere `primary: true`.

Cicli

Voce	Cosa fa
`cycles.timezone`	Il fuso con cui si decide che giorno è. Scrivilo esplicito se il server sta in un altro paese: è l'errore più comune di tutti.
`cycles.reset-at`	A che ora gira il giorno. Predefinito le 04:00.
`cycles.week-starts-on`	Quando comincia la settimana.

Mondi e anti-abuso

Voce	Cosa fa
`worlds.only` / `worlds.except`	Dove le missioni contano. `only` vince quando c'è.
`anti-abuse.ignore-afk`	Chi ha il client fermo non fa avanzare niente.
`anti-abuse.max-events-per-second`	Il tetto di eventi che possono contare in un secondo, per giocatore.
`anti-abuse.ignore-spawner-mobs`	I mob nati da uno spawner non contano.
`anti-abuse.ignore-placed-blocks`	Un blocco posato e poi rotto non conta.

Messaggi ed editor

Voce	Cosa fa
`messages.default-locale`	La lingua predefinita.
`messages.follow-client-locale`	Ogni giocatore vede i messaggi nella lingua del proprio client.
`messages.announce-progress`	L'avanzamento nella barra sopra all'inventario, non in chat.
`messages.bar.*`	Lunghezza e caratteri della barra di avanzamento.
`quests.auto-claim`	Il premio arriva da solo. Predefinito `false`, e non è pigrizia: andare a ritirare un premio è il momento in cui qualcuno apre il menu e ne comincia un'altra.
`editor.enabled`	Spegne l'editor. Le missioni continuano a funzionare: si scrivono a mano.
`metrics.*`	Le statistiche anonime di bStats. Vedi le domande frequenti.

11. Domande frequenti

Le giornaliere non si sono reimpostate

Tre cose da guardare, in quest'ordine. La prima: i cicli automatici sono **Premium**; sulla versione gratuita le missioni ci sono e funzionano ma non si rinnovano da sole. La seconda: `cycles.timezone`, se il server sta in un datacenter in un altro paese. La terza: `/sqa info` dice quanto manca alla prossima reimpostazione.

Quello che **non** può essere il problema è un server spento all'ora prevista: la reimpostazione non è un timer, è un confronto fra due nomi, e il primo giocatore che entra la mattina trova le missioni nuove.

Una missione non avanza mai

Guarda la console all'avvio: se il filtro non può corrispondere - un materiale su un obiettivo di uccisioni, un'entità che non esiste - c'è scritto lì, con il nome della missione. Poi `/sqa check`. Poi controlla `worlds.only`: se il mondo in cui giochi non è nell'elenco, niente conta.

Se l'obiettivo è sulle uccisioni e stai provando in una trappola per mob, è l'anti-abuso: i mob nati da uno spawner non contano. Si spegne con `anti-abuse.ignore-spawner-mobs: false`, o solo per quella missione con `no-spawner-mobs: false`.

`/sq` non funziona

Ce l'ha un altro plugin. Guarda la console all'avvio: c'è scritto quale etichetta è stata concessa e quali sono andate perse. `/shadowquests` risponde sempre.

Quanto pesa sul server?

Quello che serve alle missioni che hai scritto, e niente altro. Gli ascoltatori si accendono solo per i tipi di obiettivo che almeno una missione usa davvero, e si spengono quando l'ultima missione che li usava sparisce. `/sqa info` dice quali sono accesi in questo momento.

Il progresso non si scrive a ogni blocco rotto: si accumula in memoria e si scarica a intervalli, all'uscita del giocatore e allo spegnimento.

Ho una rete di server: il progresso è condiviso?

Sì, con MySQL e la sincronizzazione accesa (Premium). Il progresso non viene copiato da un server all'altro: viene **sommato dentro al database**, quindi due server che fanno avanzare la stessa missione dello stesso giocatore non si perdono niente per strada.

Un giocatore dice di non aver ricevuto un premio

Il registro c'è (`storage.history.enabled`, acceso di serie) e tiene chi ha ricevuto cosa, quando e per quale missione. È l'unica traccia che resta di una ricompensa che è un comando.

Cosa manda a bStats?

Quanti giocatori ci sono adesso, la versione di Minecraft, di Java e del sistema operativo, se il server è Premium, quante missioni ha caricate, che archivio usa e quali plugin della suite ha. **Niente sui giocatori**: nessun nome, nessun indirizzo, nessun progresso. Si spegne con `metrics.enabled: false`, oppure da `plugins/bStats/config.yml`, che vale per tutti i plugin del server insieme.

Posso usare l'editor e i file insieme?

Sì. Sono la stessa cosa: l'editor scrive normali file YAML con le normali chiavi, e una missione scritta a mano si apre nell'editor. L'unica accortezza è che l'editor riscrive il ramo della missione che stai salvando, quindi i commenti che avevi messo *dentro* a quella missione si perdono. Quelli nel resto del file restano.

Ho perso la licenza / ho cambiato server

La licenza è legata agli indirizzi. Se cambi server, `/quest license claim` con lo stesso codice: il servizio riconosce l'impronta e riemette. Se gli indirizzi non corrispondono hai sette giorni di tolleranza per sistemare, e nel frattempo tutto continua a funzionare.