SUITE SHADOW

ShadowCustomizer

Item, incantesimi, mob, blocchi e ricette che in Minecraft non esistono. Un plugin solo, una sintassi sola, una GUI sola — e tutta la suite che sa già come chiamarli.

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

1. Che cos’è ShadowCustomizer

Serve a creare contenuti che in Minecraft non esistono — item, incantesimi, mob, boss, blocchi e ricette — e a renderli usabili da tutto il resto del server.

Il problema

Chi oggi vuole item personalizzati, incantesimi e boss compra tre plugin: uno per gli item e i blocchi, uno per gli incantesimi, uno per i mob. Poi li configura in tre modi diversi, con tre sintassi diverse, tre cartelle diverse e tre GUI diverse. E spera che vadano d’accordo, perché non è detto: il boss del plugin dei mob non sa niente degli item del plugin degli item, e il premio della crate non sa niente di nessuno dei due.

Cosa fa questo

Tutte e tre le cose, con la **stessa sintassi** e la **stessa GUI**. Il clic destro di un item, l’effetto di un incantesimo e l’abilità di un boss si scrivono nello stesso modo, con gli stessi verbi: chi ha imparato a scrivere un item sa già scrivere un boss.

> **E POI C’È IL RESTO DELLA SUITE**
>
> Un boss di ShadowCustomizer può lasciare cadere un item di ShadowCustomizer, che è il premio di una crate di ShadowCrazyChest, che è l’obiettivo di una missione di ShadowQuests, che dà punti su ShadowBattlePass. Nessun concorrente può farlo, perché non possiede gli altri pezzi.

Cosa NON fa

- **Non aggiunge tipi di entità.** Un mob personalizzato è un mob vanilla vestito: statistiche, equipaggiamento, nome, comportamento, drop. Nessun plugin può fare diversamente, compresi quelli che non lo dicono.
- **Non aggiunge blocchi.** Un blocco personalizzato è uno stato del blocco nota con un modello diverso. Vale per tutti, ed è spiegato per intero al capitolo 7.
- **Non disegna niente.** I modelli 3D e il pacchetto di risorse sono l’estensione a parte, il capitolo 9. Senza, tutto funziona lo stesso e gli item si vedono come il materiale di cui sono fatti.

2. Requisiti e installazione

Server	Paper 1.26.2 o compatibile (api-version 26.2)
Java	25 o superiore
Folia	supportato
Dipendenze	nessuna
Database	nessuno da installare

1. Copia `ShadowCustomizer-1.0.0.jar` in `plugins/`.
2. Avvia il server. Al primo avvio compaiono la cartella `plugins/ShadowCustomizer/` e i file di esempio.
3. Scrivi `/scus` e guardati intorno.

> **DA DOVE COMINCIARE**
>
> Apri `items/esempi.yml`. Non è un file da cancellare: è il manuale più utile che c'è, perché ogni esempio là dentro funziona davvero. Prova `/scus item dai spada_di_rubino` e poi clicca col destro.

Le cartelle

Cartella	Cosa c'è dentro
`items/`	Gli item. Quanti file vuoi, letti in ordine di nome.
`mobs/`	I mob e i boss.
`enchants/`	Gli incantesimi. Vedi il capitolo 5: vogliono un riavvio.
`blocks/`	I blocchi.
`recipes/`	Le ricette.
`menus/`	Le GUI: righe, slot, materiali, testi. Modificabili senza ricompilare.
`lang/`	I messaggi, in italiano e in inglese.
`data/`	Il database locale. Non contiene nessuna definizione.

Puoi mettere quanti file vuoi in ognuna di quelle cartelle: vengono raccolti da soli, senza registrare niente da nessuna parte. Chi ha più di venti item li divide per tema — `armi.yml`, `consumabili.yml`, `evento-halloween.yml`.

> **UN FILE DA NON TOCCARE**
>
> `blocks-state.yml` dice quale stato del blocco nota appartiene a quale blocco personalizzato. Non va modificato e non va cancellato: cambiarlo trasforma in silenzio i blocchi già piazzati nei tuoi mondi in blocchi diversi. Il capitolo 7 spiega perché.

3. Gratuito e Premium

C'è un jar solo. Senza chiave gira la versione gratuita, con una chiave firmata gira la Premium: le funzioni a pagamento sono dentro al jar tutto il tempo, spente. Attivare è istantaneo e non scarica niente.

La versione gratuita è il prodotto intero, in piccolo

Tiene **tutto il motore**: le definizioni in YAML, la GUI, i comportamenti al clic, gli attributi, i drop, le ricette, l'API per gli altri plugin della suite. Un server piccolo ci fa un set di item a tema, tre mob e un paio di incantesimi, e non incontra mai un limite.

	Gratuito	Premium
Item	15	**illimitati**
Mob	5	**illimitati**
Incantesimi	3	**illimitati**
Blocchi	3	**illimitati**
Ricette	10	**illimitate**
Boss: barra della vita, fasi, abilità a tempo	**no**	**sì**
Incantesimi dal tavolo e dai libri	**no**	**sì**
MySQL condiviso fra più server	**no**	**sì**
API per il resto della suite	**sì**	**sì**

> **NIENTE DI COSTRUITO SI PERDE, MAI**
>
> Quello che eccede il limite **non viene cancellato**: resta nel suo file, intatto, e torna da solo appena arriva la chiave. Un mob dichiarato boss su un server gratuito compare lo stesso, con le sue statistiche e i suoi drop: quello che non ha è la barra, le fasi e le abilità.

Come si attiva

Se hai un **codice** ricevuto comprando:

```
/scus license claim SCU-XXXX-XXXX
```

Il server chiede da solo la licenza al servizio di attivazione, che gliela firma legata ai suoi indirizzi. Un secondo, nessuna persona coinvolta.

Se hai già una **chiave** in mano:

```
/scus license activate SCU1.xxxxx.yyyyy
```

Viene scritta in `license.yml`, che è un file separato dal config apposta: una richiesta di assistenza che dice «mandami il config» non deve arrivare con dentro la tua licenza. Il comando `/scus license` mostra sempre la chiave **mascherata**, anche a un operatore.

> **IL CASO PEGGIORE È TORNARE GRATUITI**
>
> Un problema di licenza non spegne mai il plugin, non lancia mai un'eccezione e non interrompe mai un'operazione a metà. Un abbonamento appena scaduto e un indirizzo che non corrisponde danno **sette giorni** di tolleranza, perché il primo è quasi sempre un rinnovo in viaggio e il secondo è molto più spesso un NAT che un ladro.

4. Gli item personalizzati

Un item si scrive così, in `items/`:

```
spada_di_rubino:
  materiale: DIAMOND_SWORD
  nome: '<gradient:#ff4d6d:#c9184a><bold>Spada di Rubino'
  descrizione:
    - '<gray>Clic destro: <white>un balzo in avanti'
  modello: 'shadow:spada_di_rubino'

  attributi:
    attack_damage: 3

  incantesimi:
    sharpness: 3
    vampirismo: 1        # anche i nostri

  comportamenti:
    clic-destro:
      attesa: 6s
      usura: 5
      azioni:
        - 'spinta: 1.6 0.55'
        - 'suono: entity.ender_dragon.flap 0.7 1.4'
```

Cosa identifica un item

L'id, e nient'altro. Sta dentro all'item, nel PersistentDataContainer: non nel nome, non nella descrizione, non nel modello.

Non è pignoleria. Il nome si cambia in un'incudine per undici livelli; la descrizione la riscrive qualunque plugin di gadget. Un item riconosciuto dal nome smette di essere sé stesso appena qualcuno lo rinomina, e chi lo fa non è un imbroglione: è uno che voleva chiamarla come gli pare. Un item riconosciuto dall'id invece sopravvive a un baule, a una enderchest, a una cassa di un altro plugin, a uno shulker dentro uno shulker, a un lancio, a un'incudine e a un aggiornamento del server con l'item nell'inventario.

> **L'ERRORE CHE FANNO TUTTI: GLI ATTRIBUTI**
>
> Nel Minecraft moderno, appena un item dichiara un modificatore di attributo, i modificatori impliciti del materiale **spariscono**. Una spada di diamante con scritto solo «+2 velocità di attacco» fa **un** danno, non sette più due.
>
> Con `attributi-base: true` — il predefinito — quelli del materiale vengono rimessi prima dei tuoi. Se hai visto una spada personalizzata fare il danno di un pugno, era questo.

I comportamenti

Trigger	Quando scatta
`clic-destro`	Tasto destro, in aria o su un blocco

Trigger	Quando scatta
`clic-sinistro`	Tasto sinistro in aria o il primo colpo su un blocco
`clic-entita`	Tasto destro su un'entità
`rompi-blocco`	Un blocco spezzato con questo in mano
`colpisci`	Un colpo andato a segno
`uccidi`	Qualcuno ucciso con questo in mano
`subisci-danno`	Danno subito con questo addosso o in mano
`in-mano`	A intervalli, finché resta in mano
`mangia`	Mangiato o bevuto
`getta`	Gettato per terra

Le opzioni di un comportamento

Chiave	Cosa fa	Predefinito
`attesa`	Quanto passa prima di poterlo rifare. **Un numero nudo sono secondi.**	0
`probabilita`	Da 0.0 a 1.0	1.0
`consuma`	Se l'item sparisce dopo	false
`usura`	Quanta durabilità costa	0
`permesso`	Un permesso da avere	nessuno
`annulla`	Se l'evento vanilla non deve avvenire. Vale solo dove c'è qualcosa da annullare: sui clic e sul getto	sì sui clic, no sugli altri
`intervallo`	Ogni quanti tick, solo per `in-mano`	20

I verbi delle azioni

Gli stessi per gli item, per gli incantesimi e per le abilità dei boss. Ogni riga è `verbo: argomenti`.

Verbo	Esempio
`messaggio`	`messaggio: <prefix><gold>Ti senti sveglio.`
`annuncio`	`annuncio: <red>Il boss si è svegliato.`
`barra`, `titolo`	`titolo: "Ce l'hai fatta" "Adesso viene il bello"`
`suono`	`suono: entity.blaze.shoot 1 1.2`

Verbo	Esempio
`particella`	`particella: FLAME 12 0.3`
`pozione`	`pozione: SPEED 200 2 su=bersaglio`
`danno`, `cura`, `sazia`	`danno: 6 su=bersaglio`
`fuoco`, `estingui`	`fuoco: 60 su=bersaglio`
`fulmine`, `esplosione`	`esplosione: 3 blocchi=false`
`spinta`	`spinta: 1.6 0.55`
`attira`, `respingi`	`respingi: 1.4 raggio=6`
`danno-vicini`	`danno-vicini: 5 raggio=6`
`comando`, `console`	`console: spawn <player>`
`dai`, `lascia`, `genera`	`lascia: rubino 1`
`soldi`, `togli-soldi`	`soldi: 50` (serve ShadowEconomy)
`esperienza`	`esperienza: 10`
`consuma`, `usura`, `rimuovi`	`usura: 5`

I SEGNAPOSTO

`<player>`, `<bersaglio>`, `<livello>`, `<x> <y> <z>`, `<mondo>`. Il nome del giocatore viene ripulito da tutto quello che in un comando diventerebbe un secondo comando: uno spazio, una virgoletta, un punto e virgola. Per i nomi normali non cambia niente.

SE CANCELLI UNA DEFINIZIONE

Gli item già in giro **non** diventano pietra e non spariscono: restano esattamente come sono, con il loro nome e i loro incantesimi, e smettono solo di avere comportamenti. Li trovi elencati in `/scus diagnostica`.

5. Gli incantesimi personalizzati

Qui gli incantesimi sono **incantesimi veri**: voci nel registro degli incantesimi del server, lo stesso in cui sta Affilatezza.

Cosa cambia rispetto agli altri plugin

I plugin di incantesimi personalizzati che esistono da anni tengono i propri in una riga di descrizione dell'item, e la rileggono a ogni colpo. Funziona, e poi non funziona: l'incudine non li sa unire, il tavolo non li sa dare, il libro non li sa contenere, e chi rinomina l'item se li porta via.

Qui invece l'incudine li unisce, il tavolo li propone, i libri li contengono, `/enchant` li conosce e qualunque altro plugin che legga gli incantesimi di un item li vede — perché non c'è niente da vedere di diverso.

> **IL PREZZO, DETTO CHIARO: SERVE UN RIAVVIO**
>
> Il registro del server si scrive **una volta sola**, all'accensione, prima che esista il mondo. Quindi aggiungere o cambiare un incantesimo vuole un riavvio, e `/scus ricarica` ricarica tutto il resto ma non questo — e te lo dice.
>
> Non è una mancanza che si possa sistemare scrivendo meglio il codice: è come funziona Minecraft, ed è anche il motivo per cui gli altri hanno scelto la descrizione. Fra «si aggiorna a caldo ma non è un vero incantesimo» e «è un vero incantesimo ma vuole un riavvio», la seconda invecchia meglio.

```
vampirismo:
  nome: '<dark_red>Vampirismo'
  descrizione: 'Ogni colpo ti ridà un po' di vita.'
  livelli: 3
  peso: 4                    # 0 = non esce dal tavolo
  costo-incudine: 4
  costo-minimo: '10 12'      # base e per livello
  costo-massimo: '60 12'
  si-mette-su:
    - '#minecraft:swords'    # i tag vanilla funzionano
    - 'trident'
  slot: [ mainhand ]
  incompatibile-con:
    - 'minecraft:fire_aspect'
  effetti:
    colpisci:
      - 'cura: 0.5*<livello>'
```

> **I NUMERI POSSONO DIPENDERE DAL LIVELLO**
>
> `3`, `1.5*<livello>`, `2+0.5*<livello>`. Una somma e un prodotto, che è tutto quello che serve davvero a un incantesimo.

Nella versione gratuita si registrano i primi tre e il loro peso vale zero: esistono, si mettono a mano o si scrivono su un item, e funzionano — semplicemente non escono dal tavolo né dai libri.

6. I mob e i boss

Un mob personalizzato è un mob vanilla vestito. Non è una limitazione di questo plugin: a nessun plugin è concesso aggiungere tipi di entità. Ed è meno di quello che sembra promettere la parola «custom» ed è tutto quello che serve: un Guardiano della Cripta con quattrocento vita, un'armatura nera, tre abilità e una barra rossa è un boss, e a nessuno importa che sotto sia uno scheletro.

```
guardiano_della_cripta:
  tipo: WITHER_SKELETON
  nome: '<dark_purple><bold>Guardiano della Cripta'
  boss: true

  statistiche:
    vita: 400
    danno: 12
    armatura: 12
    dimensione: 1.4

  brucia-al-sole: false
  equipaggiamento:
    testa: NETHERITE_HELMET
    mano:
      item: spada_di_rubino
      cade: 0.15

  barra:
    colore: PURPLE
    stile: NOTCHED_10

  abilita:
    zannata:
      ogni: 8s
      probabilita: 0.7
      raggio: 12
      azioni:
        - 'danno: 6 su=bersaglio'

  fasi:
    infuriato:
      sotto: 0.5
      colore-barra: RED
      quando-entra:
        - 'annuncio: <red>Il Guardiano si infuria!'
      abilita:
        onda:
          ogni: 12s
          serve-bersaglio: false
          azioni:
            - 'respingi: 1.4 raggio=6'
            - 'danno-vicini: 5 raggio=6'

  drop:
    - 'spada_di_rubino 1 1 1.0 solo-giocatore=true'
    - 'rubino 3 8 1.0'
```

Le fasi

Ogni fase ha una soglia (una frazione della vita massima), delle azioni che partono entrandoci, e un proprio elenco di abilità che sostituisce quello di prima. L'ordine in cui le scrivi non conta: vengono riordinate da sole.

Le fasi si attraversano **una volta sola e in discesa**: curare un boss non lo riporta alla fase precedente. Un boss che risale rifarebbe il suo monologo e riazzererebbe le attese, e la prima persona a scoprirlo lo userebbe per tenerlo fermo in eterno.

I drop

Ogni riga è `id minimo massimo probabilità`, più le precisazioni:

Precisazione	Cosa fa
`saccheggio=0.02`	Quanto sale la probabilità per livello di Saccheggio
`solo-giocatore=true`	Cade solo se a uccidere è stato un giocatore

Il saccheggio alza la **probabilità**, non la quantità: è quello che fa Minecraft con i drop rari, ed è anche quello che tiene i numeri sotto controllo. E `solo-giocatore` esiste perché senza, la prima fattoria di mob svuota l'economia del server in un pomeriggio.

> **UN SOLO TASK PER TUTTI I BOSS**
>
> I boss con abilità a tempo sono la via più rapida per far crollare i TPS, e la ragione è sempre la stessa: un task per boss. Qui su Paper c'è **un timer solo** che scorre la lista dei mob vivi. Su Folia è diverso, ed è spiegato al capitolo 12.

7. I blocchi personalizzati

Come funziona il trucco, per intero

A un plugin non è concesso aggiungere blocchi. Quello che si può fare — e che fanno tutti, compreso chi non lo dice — è prendere un blocco vanilla che ha **molti stati visivamente identici** e usarne uno per ciascun blocco nostro. Il pacchetto di risorse poi disegna ogni stato in modo diverso, e agli occhi di chi gioca sono blocchi diversi.

Qui si usa il **blocco nota**: 27 strumenti per 25 note fanno 675 combinazioni.

> **PERCHÉ NON SI USA «POWERED»**
>
> Sarebbe il doppio delle combinazioni, ma lo stato `powered` lo cambia la redstone — e con questo meccanismo lo stato **è** l'identità del blocco. Un blocco che passa da spento ad acceso perché qualcuno ha messo una leva accanto diventerebbe un altro blocco, e non ci sarebbe nessun posto in cui accorgersene. Seicentosettantacinque sono molte più di quante ne servano a chiunque.

Le quattro difese

Minecraft, di suo, cambierebbe quello stato in quattro modi. Tutti e quattro sono fermati:

Cosa succederebbe	Cosa si fa
Un blocco nota ricalcola lo strumento guardando cosa ha sotto	L'aggiornamento fisico viene annullato
Il clic destro alza la nota di uno	Il clic viene annullato (le azioni girano lo stesso)
La redstone lo fa suonare	La nota viene annullata
I pistoni lo spostano	La spinta viene annullata

> **BLOCKS-STATE.YML NON SI TOCCA E NON SI CANCELLA**
>
> Quel file dice quale stato appartiene a quale blocco. Un blocco piazzato nel mondo porta con sé il proprio stato e nient'altro: se domani lo stesso stato venisse dato a un'altra definizione, ogni blocco già piazzato in ogni mondo diventerebbe in silenzio l'altro blocco — con l'altro nome, l'altro drop e l'altro modello.
>
> Per la stessa ragione, uno stato liberato da una definizione cancellata resta occupato per sempre: si perde una combinazione su seicentosettantacinque e si guadagna che i mondi non si rovinano.

> **SE HAI ANCHE ITEMSADDER, ORAXEN O NEXO**
>
> Usano lo stesso trucco. Se due plugin scelgono lo stesso stato, i blocchi si scambiano fra loro — e non c'è nessun modo di accorgersene se non guardandoli. ShadowCustomizer te lo dice all'avvio; si sistema fissando `stato:` a mano su numeri che l'altro plugin non usa.

```yaml
minerale_di_rubino:
  nome: '<red>Minerale di Rubino'
  materiale-item: REDSTONE_ORE
  modello: 'shadow:minerale_di_rubino'
  attrezzo: piccone       # senza, si raccoglie a mani nude
  drop:
    - 'rubino 1 3 1.0'
  suono-rottura: 'block.deepslate.break'
```

`attrezzo:` serve più di quanto sembri: il blocco vero è un blocco nota, e il blocco nota si rompe con qualunque cosa. Senza quella riga, un minerale personalizzato si raccoglie a mani nude.

8. Le ricette

Ogni ingrediente e ogni risultato può essere un id nostro o un materiale vanilla. L'ordine di ricerca è quello: prima i nostri, poi il gioco. Se hai chiamato un tuo item `diamond`, quando scrivi `diamond` intendi il tuo.

Tipo	Dove
`forma`	Tavolo da lavoro, con la griglia
`libera`	Tavolo da lavoro, senza forma
`fornace`, `altoforno`, `affumicatoio`, `falo`	Le quattro cotture
`fucina`	Il tavolo da fabbro
`tagliapietre`	Il tagliapietre
`incudine`	L'incudine — vedi la nota

```
spada_di_rubino:
  tipo: forma
  forma:
    - ' R '
    - ' R '
    - ' B '
  ingredienti:
    R: rubino
    B: STICK
  risultato: spada_di_rubino
```

L'INCUDINE NON È UNA RICETTA DI MINECRAFT

L'incudine sa unire, riparare e rinominare, e basta. Una «ricetta d'incudine» è quindi il plugin che guarda i due slot e decide cosa mettere nel terzo. Per te che scrivi il file è la stessa cosa; per il server no, e per questo non compare nel libro delle ricette.

IL BUCO CHE GLI ALTRI LASCIANO APERTO

Una «Spada del Drago» costruita su un `DIAMOND_SWORD` vale, per il tavolo da lavoro, un diamante e un bastone: si può **fondere** in un lingotto, si può usare come ingrediente al posto di un item normale, e chi ha speso un mese a guadagnarsela può distruggerla con uno shift-clic.

`recipes.protect-from-vanilla` è acceso di serie e impedisce che un item personalizzato entri in una ricetta del gioco. Spegnilo solo se ti serve davvero il contrario, e sapendo cosa comporta.

9. L’estensione per i modelli 3D

ShadowCustomizerModels è il jar a parte che genera il pacchetto di risorse: i modelli, le definizioni di modello, il blockstate del blocco nota, il pack.mcmeta, lo zip e l’impronta. E soprattutto: lo **fonde** con quello che hai già.

Senza, ShadowCustomizer funziona esattamente uguale — item, incantesimi, mob, blocchi, ricette, attributi, comportamenti. Quello che cambia è che gli item si vedono come il materiale di cui sono fatti.

La fusione

È il punto dolente di ogni plugin di questo genere: chi ha già un pacchetto suo e vuole aggiungerci gli item soffre. Due pacchetti si sovrappongono su pochi file — `pack.mcmeta`, il blockstate del blocco nota — e sovrapporsi su uno di quelli significa cancellare il lavoro di qualcuno.

La regola qui è una sola, ed è scritta e detta:

> **CHI C’ERA PRIMA VINCE**
>
> Il tuo pacchetto è il tuo lavoro; il nostro è quello che si aggiunge. Dove i due dicono cose diverse sullo stesso campo, tiene il tuo e il conflitto **ti viene detto** — con il file e il nome del campo, così puoi andare a guardare. Il tuo pacchetto non viene mai modificato: si legge e si copia.

Le cartelle dell’estensione

Cartella	Cosa ci metti
`modelli/`	Come si vede ogni item e ogni blocco (i file .yml)
`texture/`	Le immagini .png, con la stessa struttura del pacchetto
`modelli-json/`	I modelli già fatti in Blockbench, se ne hai
`pacchetto/`	**Generata.** Viene svuotata a ogni generazione: non metterci niente a mano
`pacchetto.zip`	Il pacchetto compresso, pronto da servire

```
spada_di_rubino:
  tipo: item
  texture: 'item/spada_di_rubino'
  padre: 'item/handheld'   # senza, resta piatta in mano
```

Il server integrato

L’alternativa è caricare un file da decine di megabyte su un hosting e ricordarsi di rifarlo a ogni modifica. Quel giro è il motivo per cui metà dei server con contenuti personalizzati ha un pacchetto vecchio di sei mesi. Con `server.enabled: true` il pacchetto si rigenera e si serve da solo, e l’indirizzo non cambia mai.

	Gratuito	Premium
Modelli generati	10	**illimitati**
Fusione con un pacchetto esistente	**no**	**sì**
Server integrato	**no**	**sì**
Consegna automatica a chi entra	**no**	**sì**

10. Comandi

Comando principale `/shadowcustomizer`, alias `/scus` e `/custom`. Nei messaggi il comando non è mai scritto a mano: se un altro plugin si è già preso `/scus`, all'avvio viene scritto in console quale scorciatoia si è persa e i messaggi nominano quella che risponde.

Comando	Cosa fa	Permesso
`/scus`	Apre la finestra principale	`...gui`
`/scus aiuto`	L'elenco dei comandi	`...use`
`/scus item dai <id> [giocatore] [n]`	Dà un item	`...admin.give`
`/scus item lista`	Gli item, con un pulsante per prenderli	`...admin.list`
`/scus item info <id>`	File, materiale, modello, revisione	`...admin.list`
`/scus item mano`	Cos'è quello che hai in mano	`...use`
`/scus item rifai`	Rifà l'item dalla definizione	`...admin.refresh`
`/scus mob spawn <id> [n]`	Fa comparire un mob	`...admin.spawn`
`/scus mob lista`	I mob, con un pulsante per generarli	`...admin.list`
`/scus mob info <id>`	Tipo, vita, fasi, drop	`...admin.list`
`/scus mob uccidi <id>`	Uccide quelli vivi di quel tipo	`...admin.spawn`
`/scus enchant <id> [liv]`	Mette un incantesimo sull'item in mano	`...admin.enchant`
`/scus enchant lista`	Gli incantesimi, e quali aspettano un riavvio	`...admin.list`
`/scus blocco dai <id>`	Dà l'item che piazza un blocco	`...admin.give`
`/scus blocco lista`	I blocchi e lo stato che occupano	`...admin.list`
`/scus blocco dove <id>`	Dove sono i blocchi di quel tipo	`...admin.list`
`/scus ricetta`	L'elenco delle ricette	`...admin.list`
`/scus diagnostica`	Cosa non torna	`...admin.diag`
`/scus stato`	Versione, licenza, conteggi, archivio	`...admin.status`
`/scus ricarica`	Rilegge tutti i file	`...admin.reload`
`/scus license [claim|activate|refresh]`	La licenza	`...admin.license`

L’estensione

Comando	Cosa fa
`/scmodels genera`	Rigenera il pacchetto
`/scmodels manda [giocatore|tutti]`	Manda il pacchetto
`/scmodels server [accendi|spegni]`	Il server integrato
`/scmodels stato`	Com’è andata l’ultima generazione
`/scmodels ricarica`	Rilegge i file dei modelli
`/scmodels license`	La licenza dell’estensione

LE CONFERME SI CLICCANO

Ricaricare, rifare un item e uccidere dei mob non si confermano riscrivendo il comando: si clicca su `[ CONFERMA ]` o su `[ ANNULLA ]`. Il gettone dietro al pulsante è monouso e legato a chi è stato interrogato, quindi un comando copiato dalla chat non conferma l’operazione di un altro e un pulsante rimasto in cronologia non può far ripartire niente due volte.

11. Permessi

Permesso	Cosa concede	Predefinito
`shadowcustomizer.use`	Usare il comando principale	op
`shadowcustomizer.gui`	Aprire la GUI	op
`shadowcustomizer.admin.give`	Dare item e blocchi	op
`shadowcustomizer.admin.spawn`	Far comparire e togliere mob	op
`shadowcustomizer.admin.enchant`	Mettere incantesimi personalizzati	op
`shadowcustomizer.admin.list`	Vedere elenchi e informazioni	op
`shadowcustomizer.admin.refresh`	Rifare un item dalla definizione	op
`shadowcustomizer.admin.diag`	Vedere e azzerare la diagnostica	op
`shadowcustomizer.admin.status`	Vedere lo stato del plugin	op
`shadowcustomizer.admin.reload`	Ricaricare la configurazione	op
`shadowcustomizer.admin.license`	Vedere e attivare la licenza	op
`shadowcustomizer.admin`	Tutti quelli qui sopra	op
`shadowcustomizermodels.use`	Usare il comando dell'estensione	op
`shadowcustomizermodels.admin`	Generare, servire, mandare	op
`shadowcustomizermodels.admin.license`	La licenza dell'estensione	op

L'ALBERO È DICHIARATO, NON SOLO CONTROLLATO

I permessi stanno in `paper-plugin.yml` con i loro figli, quindi un amministratore a cui il plugin dei permessi dà `shadowcustomizer.admin` riceve anche tutti i figli senza doverli elencare a mano.

Una singola definizione di item può chiedere un permesso suo, scritto nel suo file con la chiave `permesso:`. Serve a fare item che solo certi gradi possono usare, e non compare in questa tabella perché lo decidi tu.

12. Dipendenze e integrazioni

Cosa serve

Niente. Zero dipendenze di terze parti sul classpath del server: quello che serve viaggia dentro al jar sotto un nome nostro. Il plugin parte e funziona su un Paper nudo.

Cosa si aggancia, se c'è

Plugin	Cosa cambia se c'è	Se manca
ShadowCustomizerModels	I modelli 3D e il pacchetto di risorse	Gli item si vedono come il loro materiale
ShadowEconomy	Le azioni `soldi` e `togli-soldi`	Quelle azioni non fanno niente
ShadowControl	La licenza può arrivare dal suo elenco	Si legge da license.yml
PlaceholderAPI	I segnaposto esterni nei testi	Restano scritti come sono

Chi ci chiama

ShadowCrazyChest, ShadowQuests, ShadowBattlePass, ShadowGames, ShadowEconomy, ShadowSpawners e ShadowBasics chiedono a ShadowCustomizer i suoi item e i suoi mob attraverso un'API pubblica registrata nel ServicesManager di Bukkit. Nessuno di loro ha bisogno di essere installato, e ognuno che c'è funziona da solo.

Folia

Supportato. Su Folia un mob appartiene alla regione in cui si trova, e nessuna abilità può toccare entità di un'altra regione da un thread sbagliato: qui ogni operazione passa dallo scheduler giusto — globale, di regione o di entità.

> **SU FOLIA C'È UN TIMER PER MOB, ED È GIUSTO COSÌ**
>
> Su Paper i boss girano in un timer solo. Su Folia quella soluzione non è possibile: un task globale che tocca la vita o il bersaglio di un mob da un altro thread non è lento, è vietato, e il server lo rifiuta. L'unico modo corretto è lo scheduler dell'entità, che è fatto apposta e che segue il mob quando cambia regione.

13. Configurazione

`config.yml` è commentato riga per riga e spiega il perché di ogni opzione. Qui c'è il riassunto delle voci che contano.

Voce	Cosa fa	Predefinito
`models.binding`	Come si legano i modelli: auto, item-model, custom-model-data, legacy-number, none	auto
`models.namespace`	Il namespace davanti a un modello scritto senza	shadow
`items.hold-period-ticks`	Ogni quanti tick gira il comportamento `in-mano`	10
`mobs.ability-period-ticks`	Ogni quanti tick il servizio dei boss guarda i mob vivi	10
`abilities.max-radius`	Il raggio massimo di un'azione ad area	32
`blocks.track-placed`	Tiene un elenco di dove sono i blocchi piazzati	true
`recipes.protect-from-vanilla`	Impedisce che un item personalizzato entri in una ricetta del gioco	true
`ui.confirm-timeout`	Quanto vale un pulsante di conferma	30s
`storage.mysql.enabled`	Usa un MySQL esterno invece del file locale (Premium)	false
`storage.fallback-to-local`	Se MySQL non risponde, riparte sul file locale	true
`license.grace-days`	Giorni di tolleranza prima di tornare gratuiti	7

IL MODELLO SI LEGA IN TRE MODI, E UNO È DEPRECATO

Il modo in cui Minecraft associa un modello a un item **è cambiato**. Il vecchio `CustomModelData` numerico funziona ancora ed è superato; quello nuovo è il componente `item_model`, che punta a un modello per nome e non ha nessun conflitto possibile con altri plugin.

`auto` sceglie da solo: se una definizione dichiara `modello:` usa il meccanismo nuovo, se dichiara `modello-numero:` usa il vecchio. Il giorno in cui il meccanismo cambierà ancora, cambierà una riga di configurazione e non le tue definizioni.

L'archivio

Nel database **non ci sono le tue definizioni**. Item, mob, incantesimi, blocchi e ricette sono file YAML e restano file YAML: si leggono, si copiano da un server all'altro, si mettono in un repository, si aprono con un editor di testo.

Nel database ci sono solo tre cose che il gioco produce mentre gira:

- dove sono i blocchi personalizzati piazzati;
- quali definizioni non esistono più ma sono ancora in circolazione;

- qualche contatore.

Di serie è un file H2 dentro alla cartella del plugin: niente da installare, nessun utente, nessuna password. MySQL serve solo a chi ha più di un server che devono vedere gli stessi dati.

14. Domande frequenti

Ho aggiunto un incantesimo e non compare

Gli incantesimi si registrano nel registro del server, che si scrive una volta sola all'accensione. Serve un **riavvio**: `/scus ricarica` ricarica tutto il resto ma non quelli, e `/scus diagnostica` ti dice esattamente quali stanno aspettando.

La mia spada personalizzata fa un danno solo

È l'errore degli attributi del capitolo 4. Metti `attributi-base: true` nella definizione, oppure dichiara tutti gli attributi tu, compresi quelli che il materiale aveva di suo.

Ho cancellato una definizione e ora i giocatori hanno item rotti

Non sono rotti: sono esattamente come prima, con il loro nome, i loro incantesimi e i loro attributi. Hanno smesso di avere comportamenti, perché la definizione che li descriveva non c'è più. `/scus diagnostica` te li elenca. Rimetti la definizione con lo stesso id e tornano a funzionare.

I miei blocchi si sono trasformati in blocchi diversi

Quasi sempre è una di queste due: hai cancellato o modificato `blocks-state.yml`, oppure hai anche ItemsAdder, Oraxen o Nexo e i due plugin si sono scelti lo stesso stato. Il capitolo 7 spiega tutte e due i casi.

Il pacchetto di risorse non si carica

Il primo posto da guardare è `pack.format` nel config dell'estensione: un formato sbagliato fa rifiutare l'**intero** pacchetto dal client, in silenzio. Il secondo è `server.public-address`: senza, il plugin scrive `127.0.0.1`, che funziona solo per te.

Gli item si vedono viola e neri

È il modo in cui Minecraft dice «manca una texture». Il modello c'è, l'immagine no: guarda in `texture/` che ci sia il file con il nome che hai scritto nella definizione. L'estensione te lo dice anche a fine generazione.

Il plugin rallenta il server?

Gli ascoltatori vengono registrati **solo se servono**. Un server che usa questo plugin per fare item decorativi non paga niente sugli eventi di combattimento; uno che non ha blocchi personalizzati non paga niente su `BlockPhysicsEvent`, che è l'evento più frequente del gioco. I due task — i boss e il comportamento `in-mano` — non partono affatto se nessuna definizione li usa.

Posso usarlo insieme a ItemsAdder o MythicMobs?

Sì. L'unica sovrapposizione vera sono i blocchi personalizzati, che usano lo stesso trucco del blocco nota: te lo diciamo all'avvio e si sistema fissando gli stati a mano. Item, incantesimi e mob non si toccano fra loro.

Dove finiscono i miei file quando aggiorno il plugin?

`items/`, `mobs/`, `enchants/`, `blocks/` e `recipes/` non vengono **mai** toccati. `config.yml` non viene mai sovrascritto: le sezioni nuove di una versione futura vengono accodate in fondo, con i loro commenti, senza toccare una riga di quello che hai scritto tu. `lang/` e `menus/` vengono sostituiti quando il jar ne porta una versione più alta, e la tua copia viene messa accanto come `.old`.

Ho un problema che non è qui

`/scus diagnostica` e `/scus stato`. Il primo dice cosa non torna — definizioni mancanti, nomi che il plugin non conosce, file scritti storti — con il percorso e cosa ci si aspettava. Il secondo dice versione, licenza, conteggi e archivio. Insieme sono quello che serve a chi ti risponde.