Configurazione JSON per le importazioni personalizzate

Questa guida spiega come creare un file JSON per importare qualunque file CSV nella raccolta dei movimenti crypto. Il JSON descrive:

- dove si trovano i dati nel CSV;
- come interpretarli (date, segni, causali);
- come trasformarli (rinomina monete, pulizia nomi, wallet).

Ogni proprietà ha un valore predefinito. Se non la inserisci nel JSON, viene usato il default indicato.

Struttura generale

```
{
"nomeExchange": "...",
"nomeWallet": "...",
"testing": false,
"separatore": ",",
"encoding": "UTF-8",
"righeIntestazione": 1,
"rigaIntestazione": 1,
"autoDetectColonne": false,
"mappaAutoDetect": { ... },
"formatoData": "yyyy-MM-dd HH:mm:ss",
"fuso": "UTC",
"consolidaRigheStessaData": false,
"tolleranzaSecondiConsolidamento": 2,
"causaliDifferite": [],
"colonne": { ... },
"mappaCausali": { ... },
"causaliChiuse": [ ... ],
"causaliUscita": [ ... ],
"causaliEntrata": [ ... ],
"ricostruisciLordoSeFeeSuMonetaUscita": false,
"ricostruisciLordoSeFeeSuMonetaEntrata": false,
"rimuoviCaseSensitive": false,
"rimuoviDaNomeMoneta": [ ... ],
"rinominaMonete": { ... },
"walletPerCausale": { ... },
"campiExtra": { ... },
"centralizzato": false,
"separatoreCausale": ".",
"causaliUppercase": false
}
```

1. Intestazione

nomeExchange

Tipo: stringa | **Default:** "" (stringa vuota)

Nome dell'exchange o della piattaforma sorgente. Se lasciato vuoto, il programma chiede interattivamente il nome in fase di importazione.

```
"nomeExchange": "Binance"
```

`nomeWallet`

Tipo: stringa | **Default:** "Principale"

Nome del wallet di destinazione. Può essere sovrascritto causale per causale tramite walletPerCausale.

```
"nomeWallet": "Spot"
```

`testing`

Tipo: booleano | **Default:** false

Se true, segnala in fase di importazione che il file di configurazione è in fase di test e l'importazione potrebbe non essere affidabile.

```
"testing": true
```

`fornitore` e `estrazione`

Tipo: stringa | **Default:** "" (stringa vuota)

Decidono **dove compare la configurazione** nelle due tendine della finestra di importazione: `fornitore` è la voce della prima tendina (l'exchange o il servizio), `estrazione` quella della seconda (quale export di quel fornitore si sta caricando).

Vanno indicati solo quando non si ricavano da soli: per un exchange basta `nomeExchange`, e per i formati che contengono i movimenti di piattaforme diverse (CoinTracking, Tatax) basta la parola nel nome del file. Se mancano entrambi si usa il nome del file di configurazione. `estrazione` va scritta **senza ripetere** il nome del fornitore, che è già nella prima tendina.

```
"fornitore": "OKX",
"estrazione": "Funding"
```

2. Struttura fisica del CSV

`separatore`

Tipo: stringa | **Default:** ","

Carattere che divide le colonne nel CSV.

Valore	Quando usarlo	
","	Standard internazionale, default.	
";"	Export da Excel con impostazioni europee.	
"\t"	File TSV (tab-separated).	
"\\	"	Caso raro, usato da alcuni exchange.

```
"separatore": ";"
```

`encoding`

Tipo: stringa | **Default:** "UTF-8"

Usa "UTF-8" nella maggior parte dei casi. Se i caratteri speciali risultano corrotti, prova "ISO-8859-1" oppure "Windows-1252".

`righeIntestazione`

Tipo: numero intero | **Default:** 1

Quante righe iniziali del CSV vanno saltate prima dei dati. Se il CSV non ha intestazione, imposta 0.

`rigaIntestazione`

Tipo: numero intero | **Default:** 1

Quale riga (indice 1-based) contiene i nomi delle colonne. Usata solo se autoDetectColonne è true.

autoDetectColonne e mappaAutoDetect **autoDetectColonne** | **Tipo:** booleano | **Default:** false

Se true, legge i nomi delle colonne dalla riga di intestazione e assegna automaticamente gli indici tramite mappaAutoDetect.

mappaAutoDetect | **Tipo:** oggetto chiave-valore

Mappa il nome dell'intestazione CSV al nome del campo logico. Valori validi: data, causale, moneta, quantita, segno, valoreEuro, prezzo, monetaFee, quantitaFee, idTransazione, wallet.

```
"autoDetectColonne": true,
"mappaAutoDetect": {
"Timestamp": "data",
"Type": "causale",
"Asset": "moneta",
"Amount": "quantita",
"Fee Amount": "quantitaFee",
"Fee Asset": "monetaFee",
"TxID": "idTransazione"
}
```

`formatoData`

Tipo: stringa | **Default:** "yyyy-MM-dd HH:mm:ss"

Formato della data/ora nel CSV, con i token Java di DateTimeFormatter.

Token	Significato	Esempio
yyyy	Anno a 4 cifre	2025
yy	Anno a 2 cifre	25
MM	Mese a 2 cifre	06

dd	Giorno a 2 cifre	17
HH	Ora 0-23	18
mm	Minuti	39
ss	Secondi	10

```
"formatoData": "dd.MM.yyyy HH:mm:ss"
"formatoData": "MM/dd/yy HH:mm"
```

fuso

Tipo: stringa | **Default:** "UTC"

Fuso orario delle date nel CSV. Vengono convertite nel fuso locale dell'app.

```
"fuso": "UTC"
"fuso": "Europe/Rome"
"fuso": "UTC+2"
```

3. Colonne del CSV

Mappa i campi logici agli indici numerici delle colonne del CSV, contando da 0. Usa -1 per indicare che una colonna non è presente.

```
"colonne": {
"data": 0,
"causale": 1,
"moneta": 2,
"quantita": 3,
"segno": -1,
"valoreEuro": -1,
"prezzo": -1,
"monetaFee": -1,
"quantitaFee": -1,
"idTransazione": -1,
"wallet": -1,
"monetaUscita": -1,
"quantitaUscita":-1,
"causale2": -1,
"causale3": -1
}
```

Come trovare l'indice di una colonna

Apri il CSV, guarda la riga di intestazione e conta le colonne partendo da 0.

```
"Timestamp","Tipo","Asset","Quantita","Valore EUR"
0 1 2 3 4
```

Quindi: "data": 0, "causale": 1, "moneta": 2 e così via.

Campi disponibili

Campo	Descrizione
data	Obbligatorio - data e ora del movimento.
causale	Tipo di operazione nel CSV (Deposit, Trade, Staking...).
moneta	Simbolo del token (BTC, ETH...).
quantita	Quantità del movimento (negativa=uscita, positiva=entrata).
segno	Colonna separata che indica il segno (+ o -).
valoreEuro	Controvalore in euro già calcolato nel CSV.
prezzo	Prezzo unitario del token al momento dell'operazione.
monetaFee	Simbolo della moneta usata per la commissione.
quantitaFee	Quantità della commissione pagata.
idTransazione	ID univoco per raggruppare righe correlate dello stesso trade.
wallet	Nome del wallet, se variabile riga per riga.
monetaUscita	Moneta in uscita (CSV con entrata e uscita sulla stessa riga).
quantitaUscita	Quantita in uscita (CSV con entrata e uscita sulla stessa riga).

Causale composita

Alcuni CSV distribuiscono il tipo di operazione su più colonne. Si possono combinare fino a 3 colonne in una causale composita concatenate da separatoreCausale.

Parametro	Default	Descrizione
causale2 (in colonne)	-1	Indice della seconda colonna causale.
causale3 (in colonne)	-1	Indice della terza colonna causale.
separatoreCausale	"."	Carattere di giunzione tra le parti.
causaliUppercase	false	Se true converte ogni parte in maiuscolo prima di concatenare.

Esempio: causale=1 ("Trade"), causale2=2 ("Buy"), separatoreCausale="." produce la chiave "Trade.Buy" da usare in mappaCausali.

```
"colonne": { "causale": 1, "causale2": 2 },
"separatoreCausale": ".",
"mappaCausali": { "Trade.Buy": "SCAMBIO CRYPTO-CRYPTO" }
```

Entrata e uscita sulla stessa riga

Alcuni CSV (es. Koinly, CoinTracking) riportano l'intero scambio su una sola riga. In quel caso moneta/quantita = lato entrata, monetaUscita/quantitaUscita = lato uscita.

```
"colonne": {
"data": 8, "causale": 0,
"moneta": 2, "quantita": 1,
"monetaUscita": 4, "quantitaUscita": 3,
"monetaFee": 6, "quantitaFee": 5
}
```

`idGruppo` (nella sezione colonne)

Tipo: numero intero | **Default:** -1 (assente)

Colonna su cui raggruppare le righe **quando non coincide con quella dell'identificativo**. I due ruoli sono distinti: `idTransazione` dice *chi è* la riga — finisce nel movimento ed è ciò su cui lavora il controllo dei duplicati — mentre `idGruppo` dice *con chi sta*.

Nell'export di trading di OKX, ad esempio, ogni gamba e ogni esecuzione parziale hanno un `id` diverso ma condividono l'`Order id`: usare il primo anche per raggruppare spezzerebbe ogni scambio in gambe isolate. Quando `idGruppo` non è indicata si raggruppa come sempre sull'identificativo, quindi le configurazioni che non la usano non cambiano comportamento.

Quando `idGruppo` è presente, il file viene inoltre **ordinato** per quel campo (e poi per `idTransazione`) prima dell'importazione, perché il raggruppamento lavora su righe consecutive e gli export intercalano le gambe di ordini diversi.

```
"colonne": { "idTransazione": 0, "idGruppo": 1 }
```

4. Mappatura delle causali

`mappaCausali`

Tipo: oggetto chiave-valore

Traduce le causali originali del CSV nelle tipologie interne dell'applicazione. Ogni causale deve avere una voce qui, altrimenti la riga viene scartata. La ricerca è case insensitive per default.

```
"mappaCausali": {
"Deposit": "DEPOSITO-CRYPTO",
"Withdrawal": "PRELIEVO-CRYPTO",
"Trade": "SCAMBIO CRYPTO-CRYPTO",
"Staking Rewards": "STAKING REWARDS",
"Commission": "COMMISSIONI",
"earn":"EARN",
"cashback":"CASHBACK"
}
```

Per ignorare righe senza contarle come scarti, mappa su "IGNORA":

```
"mappaCausali": { "Internal Transfer": "IGNORA" }
```

`causaliChiuse`

Tipo: array di stringhe (tipologie interne)

Tipologie interne che devono essere trattate come movimenti singoli anche se condividono l'ID transazione con altre righe. Usa per staking, cashback, commissioni, depositi e prelievi.

```
"causaliChiuse": [
"DEPOSITO-CRYPTO","PRELIEVO-CRYPTO",
"STAKING REWARDS","CASHBACK","COMMISSIONI"
```

]

5. Gestione del segno

Per default il segno viene letto dalla quantità (negativa = uscita, positiva = entrata). Se la quantità è sempre positiva e il verso si deduce dalla causale, usa causaliUscita e causaliEntrata.

`causaliUscita`

Tipo: array di causali CSV originali

Le righe con queste causali avranno la quantità forzata negativa.

```
"causaliUscita": ["Withdrawal","Trade Sell","Fee"]
```

`causaliEntrata`

Tipo: array di causali CSV originali

Le righe con queste causali avranno la quantità forzata positiva.

```
"causaliEntrata": ["Deposit","Trade Buy","Staking Rewards","Cashback"]
```

Esempio pratico

```
"Data","Tipo","Moneta","Quantita"
"2025-01-15","Deposito","BTC","0.5"
"2025-01-16","Prelievo","ETH","1.2"
"causaliUscita": ["Prelievo"],
"causaliEntrata": ["Deposito","Staking","Cashback"]
```

6. Consolidamento righe correlate

Alcuni exchange suddividono una singola operazione in più righe CSV (es. riga per moneta venduta + riga per moneta acquistata). Le opzioni seguenti permettono di raggrupparle.

`consolidaRigheStessaData`

Tipo: booleano | **Default:** false

- false: ogni riga è un movimento indipendente.

- true: le righe con stesso ID transazione o timestamp vicino vengono raggruppate.

```
"consolidaRigheStessaData": true
```

`tolleranzaSecondiConsolidamento`

Tipo: numero intero | **Default:** 2

Usato solo se `consolidaRigheStessaData` è true. Quanti secondi di differenza sono tollerati tra due righe perché vengano considerate parte della stessa operazione.

```
"tolleranzaSecondiConsolidamento": 5
```

`causaliDifferite`

Tipo: array di causali CSV originali

Se specificato, la tolleranza temporale viene applicata solo ai gruppi che contengono almeno una riga con queste causali; per le altre viene richiesto timestamp identico.

```
"causaliDifferite": ["Trade","Swap"]
```

`idTransazione` (nella sezione colonne)

Se il CSV ha una colonna con un ID univoco per transazione, indicarla consente di raggruppare righe anche se i timestamp differiscono oltre la tolleranza.

```
"colonne": { "idTransazione": 5 }
```

7. Gestione commissioni

`ricostruisciLordoSeFeeSuMonetaUscita`

Tipo: booleano | **Default:** false

Controlla se, quando la commissione è nella stessa moneta dell'uscita, il sistema deve ricostruire il lordo del movimento principale.

- true: ricostruisce il lordo e genera anche il movimento commissione separato.
- false: non modifica il movimento principale, genera solo il movimento commissione.

Esempio: uscita -500 USDC e fee 1 USDC. Con true: movimento principale -499 USDC + movimento commissione -1 USDC. Usalo quando la quantità nel CSV è già comprensiva della fee (riga unica).

```
"ricostruisciLordoSeFeeSuMonetaUscita": true
```

`ricostruisciLordoSeFeeSuMonetaEntrata`

Tipo: booleano | **Default:** false

Controlla se, quando la commissione è nella stessa moneta dell'entrata, il sistema deve ricostruire il lordo del movimento principale.

- true: ricostruisce il lordo e genera anche il movimento commissione separato.
- false: non modifica il movimento principale, genera solo il movimento commissione.

Esempio: entrata 500 USDC e fee 1 USDC. Con true: movimento principale 501 USDC + movimento commissione -1 USDC. Usalo quando la quantità nel CSV è al netto della fee (riga unica).

```
"ricostruisciLordoSeFeeSuMonetaEntrata": true
```

8. Pulizia e normalizzazione nomi moneta

`rimuoviDaNomeMoneta`

Tipo: array di stringhe

Rimuove testi indesiderati dal nome del token. Supporta la sintassi ? per troncare tutto ciò che viene dopo o prima.

Sintassi	Effetto su BTC.STAKING@CRYPTO.COM	Risultato
.STAKING?	Rimuove .STAKING e tutto quello che segue	BTC
?.STAKING	Rimuove .STAKING e tutto quello che precede	@CRYPTO.COM
.STAKING	Rimuove solo la parola esatta	BTC@CRYPTO.COM

Le regole vengono applicate in sequenza. Prima quelle più ampie (con ?).

```
"rimuoviDaNomeMoneta": [".STAKING?",".EARN?",".LOCKED?","@CRYPTO.COM"]
```

`rimuoviCaseSensitive`

Tipo: booleano | **Default:** false

- false: la ricerca ignora maiuscole e minuscole.
- true: la ricerca e case sensitive esatta.

`rinominaMonete`

Tipo: oggetto chiave-valore

Rinomina un simbolo moneta. La rinomina avviene dopo la pulizia con rimuoviDaNomeMoneta.

```
"rinominaMonete": { "IOTA": "MIOTA", "LUNA2": "LUNA", "WBTC": "BTC" }
```

9. Wallet per causale

`walletPerCausale`

Tipo: oggetto chiave-valore

Permette di assegnare un wallet diverso da nomeWallet per specifiche causali. La chiave è la causale

originale del CSV.

```
"walletPerCausale": {
"Staking Rewards": "Staking", "Earn": "Earn", "Lockup": "Locked"
}
```

10. Campi extra

`campiExtra`

Tipo: oggetto indiceMovimento -> indiceColonnaCSV

Copia il contenuto di una colonna CSV in un campo specifico del movimento. Funzione avanzata e raramente necessaria.

```
"campiExtra": { "7": 9 }
```

11. Centralizzato

`centralizzato`

Tipo: booleano | **Default:** false

Indica che questo file è gestito centralmente dal repository ufficiale. All'avvio del programma, quando vengono controllati i file di configurazione da GitHub, se un file locale ha centralizzato: true e non è più presente nel repository, viene automaticamente eliminato.

I file creati localmente dall'utente non devono avere centralizzato: true, altrimenti potrebbero essere cancellati involontariamente.

```
"centralizzato": true
```

Esempi completi

Esempio 1: Binance Spot - righe separate per ogni lato dello scambio CSV
"Date(UTC)","OrderNo","Pair","Type","Filled","Total","Fee","Fee Coin"

```
"2025-03-10 14:22:01","123456","BTCUSDT","BUY","0.01 BTC","620.50 USDT","0.00001","BTC"
"2025-03-10 14:22:01","123456","BTCUSDT","SELL","620.50 USDT","","0.62","USDT"
```

JSON:

```
{
"nomeExchange": "Binance", "nomeWallet": "Principale",
"separatore": ",", "formatoData": "yyyy-MM-dd HH:mm:ss", "fuso": "UTC",
"consolidaRigheStessaData": true, "tolleranzaSecondiConsolidamento": 2,
"colonne": {
"data": 0, "idTransazione": 1, "causale": 3,
"moneta": 2, "quantita": 4, "quantitaFee": 6, "monetaFee": 7
},
"mappaCausali": { "BUY": "SCAMBIO CRYPTO-CRYPTO", "SELL": "SCAMBIO CRYPTO-CRYPTO" }
}
```

Esempio 2: Riga singola per movimento - es. Tatax

CSV:

```
"Data","Tipo","Asset","Quantita"
"2025-09-15 10:00:00","Deposito","BTC","0.5"
"2025-09-16 11:30:00","Staking","BTC.STAKING@CRYPTO.COM","0.0001"
"2025-09-17 09:00:00","Prelievo","ETH","0.1"
```

JSON:

```
{
"nomeExchange": "Crypto.com", "nomeWallet": "Principale",
"separatore": ",", "formatoData": "yyyy-MM-dd HH:mm:ss", "fuso": "UTC",
"consolidaRigheStessaData": false,
"colonne": { "data": 0, "causale": 1, "moneta": 2, "quantita": 3 },
"mappaCausali": {
"Deposito": "DEPOSITO-CRYPTO", "Prelievo": "PRELIEVO-CRYPTO",
"Staking": "STAKING REWARDS"
},
"causaliChiuse": ["DEPOSITO-CRYPTO","PRELIEVO-CRYPTO","STAKING REWARDS"],
"causaliEntrata": ["Deposito","Staking"],
"causaliUscita": ["Prelievo"],
"rimuoviDaNomeMoneta": [".STAKING?",".EARN?","@CRYPTO.COM"],
"rimuoviCaseSensitive": false
}
```

Esempio 3: Scambio su riga singola - es. CoinTracking

CSV:

```
"Tipo","Acquisto","Cur.","Vendita","Cur.","Fee","Cur.Fee","Exchange","Data"
"Operazione","2132","CRO","487.04","USDC","1.18","USDC","Crypto.com","17.09.2025 18:39:10"
"Deposito","89.4","CRO","","","","","Crypto.com","28.08.2025 07:38:42"
"Prelievo","","","24.32","USDC","","","Crypto.com","03.09.2025 18:11:09"
```

JSON:

```
{
"nomeExchange": "Crypto.com Exchange", "nomeWallet": "Principale",
"separatore": ",", "formatoData": "dd.MM.yyyy HH:mm:ss", "fuso": "UTC",
"colonne": {
"data": 8, "causale": 0, "moneta": 2, "quantita": 1,
"monetaUscita": 4, "quantitaUscita": 3, "quantitaFee": 5, "monetaFee": 6
},
"mappaCausali": {
"Deposito": "DEPOSITO-CRYPTO", "Prelievo": "PRELIEVO-CRYPTO",
"Operazione": "SCAMBIO CRYPTO-CRYPTO"
},
"ricostruisciLordoSeFeeSuMonetaUscita": false,
"ricostruisciLordoSeFeeSuMonetaEntrata": true
}
```

Esempio 4: Causale composita su più colonne

CSV:

```
"Date","Category","SubType","Amount","Currency"
"2025-01-10","Trade","Buy","0.005","BTC"
"2025-01-10","Trade","Sell","200","USDT"
"2025-01-11","Earn","Staking","0.0001","ETH"
```

JSON:

```
{
"nomeExchange": "Exchange XYZ", "separatore": ",",
"formatoData": "yyyy-MM-dd", "fuso": "UTC",
"consolidaRigheStessaData": true,
"colonne": { "data": 0, "moneta": 4, "quantita": 3, "causale": 1, "causale2": 2 },
"separatoreCausale": ".", "causaliUppercase": false,
"mappaCausali": {
"Trade.Buy": "SCAMBIO CRYPTO-CRYPTO",
"Trade.Sell": "SCAMBIO CRYPTO-CRYPTO",
"Earn.Staking": "STAKING REWARDS"
},
"causaliChiuse": ["STAKING REWARDS"],
"causaliEntrata": [], "causaliUscita": []
}
```

Checklist rapida

- Apri il CSV e conta le colonne partendo da 0.
- Identifica data, causale, moneta e quantità; compila la sezione colonne.
- Elenca tutte le causali presenti nel CSV e compila mappaCausali.
- Se le quantità sono sempre positive, compila causaliUscita e causaliEntrata.
- Se ogni riga è un movimento indipendente usa consolidaRigheStessaData: false; se più righe appartengono allo stesso trade usa true.
- Se i nomi moneta contengono suffissi, compila rimuoviDaNomeMoneta.
- Se ci sono commissioni, mappa monetaFee e quantitaFee; se la quantità è già comprensiva della fee abilita il flag ricostruisciLordo appropriato.
- Se alcuni movimenti devono andare su wallet separati, compila walletPerCausale.
- Se le intestazioni CSV variano di versione in versione, usa autoDetectColonne con mappaAutoDetect.
- Se il tipo di operazione è su più colonne, usa causale2/causale3 con separatoreCausale.

Dove mettere il file

Nella cartella di lavoro del programma si trovano due cartelle di configurazione:

- `config/import/` — è la cartella attuale, sincronizzata con il repository ufficiale: qui arrivano le configurazioni distribuite con il programma;
- `ImportConfig/` — la cartella storica, mantenuta per compatibilità con le installazioni precedenti. Da qui vengono letti soltanto i file **non** marcati `"centralizzato": true`, cioè quelli scritti dall'utente.

Un file JSON messo in una di queste due cartelle compare nella finestra di importazione insieme agli import nativi, ordinato per fornitore ed estrazione (il vecchio prefisso `[JSON]` non c'è più: la distinzione è nei campi, non nell'etichetta).

I file creati personalmente vanno lasciati **senza** `centralizzato` (o con `"centralizzato": false`): non verranno mai eliminati automaticamente dall'aggiornamento dal repository remoto.