OCM-R: OneChronos Rates Technical Reference Guide



Version 1

September 25, 2026

Contents

1. Introduction 2
2. Encodings 2
   - 2.1. Protocol Versioning 3
   - 2.2. MultiLeg EB Example 3
     - 2.2.1. J-Encoding 3
     - 2.2.2. F-Encoding 4
   - 2.3. Outright EB Example 8
     - 2.3.1. J-Encoding 9
     - 2.3.2. F-Encoding 9
   - 2.4. JSON Schema 10
3. EB Lifecycle 11
4. EB Identifiers 12
5. FIX Specification 13
   - 5.1. Message Catalog 13
     - 5.1.1. Subscriber to OCM-R 13
     - 5.1.2. OCM-R to Subscriber 13
   - 5.2. EB Cancellation 37

© 2026 OneChronos Markets LLC

5.3. Novation Risk Heartbeats 39
6. The CERT Self-Certification Environment 41
6.1. FIX Sample Client 41
6.2. Install uv 42
6.3. Download the Client 42
6.4. Configure the Session 42
6.5. Run the Samples 43
7. Type Index 44
7.1. FIX Types 44
7.2. JSON Types 44
8. Disclaimers 45

1. Introduction

OneChronos Rates ("OCM-R" or the "venue") is a trading venue for U.S. Treasury securities on the OneChronos Fixed Income ATS. Subscribers submit trading intent to OCM-R using the FIX protocol. This document specifies that protocol and describes the self-certification environment used by Subscribers to test their FIX implementations.

With OCM-R, trading intent is represented by an **Expressive Bid ("EB")**. An EB may package trading interest in multiple instruments together with constraints that apply to the resulting basket of fills. An arbitrary multi-instrument EB is called a **MultiLeg**. To support the common use case of trading a single instrument, the protocol also offers a convenience message type called **Outright**.

2. Encodings

An EB is submitted using one of two FIX encodings:

J-encoding (`NewOrderSingle`, `MsgType=D`): One JSON object in `EBJson` (tag `20050`) carries the complete versioned EB, either a MultiLeg EB or an Outright EB. The required top-level key names the schema revision, for example `{ "v1": { ... } }`. A missing or unrecognized version is rejected because there is no default version. The full schema follows.

F-encoding (`EBContext`, `MsgType=U9`, plus leg `NewOrderSingle` messages, `MsgType=D`): The EB is represented across FIX messages and tags. For a MultiLeg EB, `EBContext` carries `EBVersion` (tag `20100`) and the EB-level fields. Each leg is a separate `NewOrderSingle` referenced by its `ClOrdID`. For an Outright EB, one `NewOrderSingle` carries `EBVersion` and the complete EB. The venue assembles a MultiLeg EB before validation.

© 2026 OneChronos Markets LLC

2.1. Protocol Versioning

Both J-encoding and F-encoding are versioned protocols. This guide defines EB protocol version 1 only. In the J-encoding, the required top-level key in the JSON value carried by the `EBJson` tag (`20050`) is `"v1"`. In the F-encoding, the version is specified once, as `EBVersion` (`20100`):

- for a MultiLeg EB, in the `EBContext` (`MsgType=U9`) message.
- for an Outright EB, in the `NewOrderSingle` (`MsgType=D`) message.

The leg messages of an F-encoded EB do not specify `EBVersion`. Currently, the only valid `EBVersion` value is the integer `1`; a missing tag or any other value is invalid.

2.2. MultiLeg EB Example

To illustrate both encoding options, first consider the following MultiLeg example:

Instrument	DV01	Side	Max Qty, $q_i^{\max}$, \$1000	Initial Position, q_i^0, \$1000
A	1.23	Buy	20	-3
B	4.56	Sell	10	—
C	7.89	Buy:Sell	25:30	—
D	0.66	—	—	22
E	0.12	Buy	7	—

basket neutrality tolerance, psi	0.05
yield of A	$y_{\mathrm{A}} \geq 3.9\%$
A:C yield spread	$y_{\mathrm{A}} - y_{\mathrm{C(Sell)}} \geq 0.1\%$
B:E yield spread	$-y_{\mathrm{B}} + y_{\mathrm{E}} \geq -0.07\%$

2.2.1. J-Encoding

The J-encoding carries the MultiLeg EB in a `NewOrderSingle` (`MsgType=D`). Using JSON, the EB is represented as follows:

```
{
  "v1": {
    "multi_leg": {
      "yield_constraints": [
        {
          "terms": {
            "A": {
              "alpha": 1
            }
          },
          "lower_bound": 3.9
        },
        {
          "terms": {
            "C": {
              "side": "sell",
              "alpha": -1
            },
```

© 2026 OneChronos Markets LLC

```
            "A": {
              "alpha": 1
            }
          },
          "lower_bound": 0.1
        },
        {
          "terms": {
            "E": {
              "alpha": 1
            },
            "B": {
              "alpha": -1
            }
          },
          "lower_bound": -0.07
        }
      ],
      "partial_fill_behavior": "rest_with_initial_qty",
      "neutrality_tolerance": 0.05,
      "instruments": {
        "E": {
          "max_buy_qty": 7000,
          "DV01": 0.12
        },
        "D": {
          "initial_qty": 22000,
          "DV01": 0.66
        },
        "C": {
          "max_sell_qty": 30000,
          "max_buy_qty": 25000,
          "DV01": 7.89
        },
        "B": {
          "max_sell_qty": 10000,
          "DV01": 4.56
        },
        "A": {
          "max_buy_qty": 20000,
          "initial_qty": -3000,
          "DV01": 1.23
        }
      },
      "id": "EB-001"
    }
  }
}
```


The J-encoding of this EB consists of just one message, where the entire JSON string above is the value of FIX tag `20050`. The EB names itself with the `"id"` inside that payload, so the message carries no `ClOrdID` (tag `11`) (`D18`):

Tag	Description	Value
	J-Encoding NewOrderSingle (`MsgType=D`)	
8	FIXVersion	FIX.4.4
35	MsgType	D
34	MessageSequenceNum	1234
49	SenderCompID	TRDR
52	SendTime	20260413-16:23:46.831
56	TargetCompID	OCMR
20050	EBJson	{"v1": {"multi_leg": {"yield_constraints": [{"terms": {"A": {"alpha": 1}}, "lower_bound": 3.9}, {"terms": {"C": {"side": "sell", "alpha": -1}, "A": {"alpha": 1}}, "lower_bound": 0.1}, {"terms": {"E": {"alpha": 1}, "B": {"alpha": -1}}, "lower_bound": -0.07}], "partial_fill_behavior": "rest_with_initial_qty", "neutrality_tolerance": 0.05, "instruments": {"E": {"max_buy_qty": 7000, "DV01": 0.12}, "D": {"initial_qty": 22000, "DV01": 0.66}, "C": {"max_sell_qty": 30000, "max_buy_qty": 25000, "DV01": 7.89}, "B": {"max_sell_qty": 10000, "DV01": 4.56}, "A": {"max_buy_qty": 20000, "initial_qty": -3000, "DV01": 1.23}}, "id": "EB-001"}}}

2.2.2. F-Encoding

The F-encoding of the example above consists of six FIX messages: five leg NewOrderSingle messages and one EBContext message. Each leg's ClOrdID (11) is repeated as a LegID (20001) in the ordered NumLegs (20010) group. That identifier, rather than message arrival order, links the leg to its containing EB; the DV01 and InitialQty fields following a LegID belong to that group entry. TermSecurityID (20024) similarly refers to a leg's SecurityID (48). The linking rows are highlighted in the complete message sequence below.

Leg Message Field	Links To	EBContext Field
ClOrdID (11)	→	NumLegs[*].LegID (20001)
SecurityID (48)	→	TermSecurityID (20024)

OCM-R holds an incomplete F-encoded EB for an assembly window of 5 seconds, during which the venue waits to receive the complete linked message set. If the leg messages arrive but the EBContext does not, the venue reports A11. If the EBContext arrives but one or more referenced legs do not, it likewise reports A10. Only a complete linked set of legs becomes an EB and then proceeds to additional semantic validation en route to the OCM-R book.

An EB may declare its lifetime using standard TimeInForce (59), carried in the same message that carries EBVersion: the EBContext for a MultiLeg EB, the NewOrderSingle for an Outright EB. Day (0), the default, rests the EB until it fully fills or is canceled. ImmediateOrCancel (3) limits it to exactly one auction.

Two custom fields mirror the J-encoding's in_force persistence field:

- InForceAuctions (20033) specifies a lifetime in terms of the number of auctions, or
- InForceSeconds (20034) specifies a lifetime in terms of seconds measured from acknowledgment.

Each must be at least 1. TimeInForce=ImmediateOrCancel is equivalent to InForceAuctions=1. The three lifetime fields are mutually exclusive; supplying more than one is an error. If the specified InForceSeconds interval expires during an accumulation phase, the EB is canceled upon expiry. If it expires during a matching phase, any residual interest is canceled after that phase completes.

In this example, o:11 through o:15 are illustrative ClOrdID values. Each value first appears on a leg message and is then repeated verbatim as the corresponding LegID in the EBContext. Highlighted rows in the table mark these linkage fields and their repeated values.

Message	Tag	Description	Value
Message 1: Leg A (MsgType=D)			
1	8	FIXVersion	FIX.4.4
1	35	MsgType	D
1	34	MessageSequenceNum	1234
1	49	SenderCompID	TRDR
1	52	SendTime	20260413-16:23:47.831
1	56	TargetCompID	OCMR

© 2026 OneChronos Markets LLC

1	11	ClOrdID	o:11
1	48	SecurityID	A
1	22	SecurityIDSource	1
1	54	Side	1
1	38	OrderQty	20000
Message 2: Leg B (MsgType=D)			
2	8	FIXVersion	FIX.4.4
2	35	MsgType	D
2	34	MessageSequenceNum	1235
2	49	SenderCompID	TRDR
2	52	SendTime	20260413-16:23:48.831
2	56	TargetCompID	OCMR
2	11	ClOrdID	o:12
2	48	SecurityID	B
2	22	SecurityIDSource	1
2	54	Side	2
2	38	OrderQty	10000
Message 3: Leg C (MsgType=D)			
3	8	FIXVersion	FIX.4.4
3	35	MsgType	D
3	34	MessageSequenceNum	1236
3	49	SenderCompID	TRDR
3	52	SendTime	20260413-16:23:49.831
3	56	TargetCompID	OCMR
3	11	ClOrdID	o:13
3	48	SecurityID	C
3	22	SecurityIDSource	1
3	54	Side	U
3	38	OrderQty	25000
3	20004	OtherOrderQty	30000
Message 4: Leg D (MsgType=D)			
4	8	FIXVersion	FIX.4.4
4	35	MsgType	D
4	34	MessageSequenceNum	1237
4	49	SenderCompID	TRDR
4	52	SendTime	20260413-16:23:50.831

© 2026 OneChronos Markets LLC

4	56	TargetCompID	OCMR
4	11	ClOrdID	o:14
4	48	SecurityID	D
4	22	SecurityIDSource	1
		Message 5: Leg E (MsgType=D)	
5	8	FIXVersion	FIX.4.4
5	35	MsgType	D
5	34	MessageSequenceNum	1238
5	49	SenderCompID	TRDR
5	52	SendTime	20260413-16:23:51.831
5	56	TargetCompID	OCMR
5	11	ClOrdID	o:15
5	48	SecurityID	E
5	22	SecurityIDSource	1
5	54	Side	1
5	38	OrderQty	7000
		Message 6: EBContext Basket (MsgType=U9)	
6	8	FIXVersion	FIX.4.4
6	35	MsgType	U9
6	34	MessageSequenceNum	1239
6	49	SenderCompID	TRDR
6	52	SendTime	20260413-16:23:48.336
6	56	TargetCompID	OCMR
6	20100	EBVersion	1
6	20000	EBID	EB-001
6	20010	NumLegs	5
6	20012	NeutralityTolerance	0.050000
6	20001	LegID	o:11
6	20005	DV01	1.230000
6	20006	InitialQty	-3000
6	20001	LegID	o:12
6	20005	DV01	4.560000
6	20001	LegID	o:13
6	20005	DV01	7.890000
6	20001	LegID	o:14
6	20005	DV01	0.660000

© 2026 OneChronos Markets LLC

6	20006	InitialQty	22000
6	20001	LegID	o:15
6	20005	DV01	0.120000
6	20020	NumYieldConstraints	3
6	20021	ConstraintID	1
6	20022	LowerBound	3.900000
6	20023	NumConstraintTerms	1
6	20024	TermSecurityID	A
6	20025	TermAlpha	1.000000
6	20021	ConstraintID	2
6	20022	LowerBound	0.100000
6	20023	NumConstraintTerms	2
6	20024	TermSecurityID	A
6	20025	TermAlpha	1.000000
6	20024	TermSecurityID	C
6	20025	TermAlpha	-1.000000
6	20026	TermSide	2
6	20021	ConstraintID	3
6	20022	LowerBound	-0.070000
6	20023	NumConstraintTerms	2
6	20024	TermSecurityID	E
6	20025	TermAlpha	1.000000
6	20024	TermSecurityID	B
6	20025	TermAlpha	-1.000000
6	20030	PartialFillBehavior	2

`OrderQty` and `OtherOrderQty` are integer amounts of part notional, denominated in U.S. dollars. `InitialQty` is a signed floating-point amount in U.S. dollars; it is optional, and an explicit zero denotes a zero initial position. `DV01` is optional unless `NeutralityTolerance` is supplied. When neutrality tolerance is supplied, every leg must provide `DV01` (`C15`), and at least one leg must have a non-zero value (`C05`). Regardless of neutrality, each leg must provide at least one of `OrderQty`, `OtherOrderQty`, `InitialQty`, or `DV01` (`C14`).

2.3. Outright EB Example

To support the common use case of trading a single instrument, the protocol offers the convenience message type **Outright**. An Outright EB expresses interest in a single instrument with a size and

© 2026 OneChronos Markets LLC

a limit, which represents either a clean price or a yield. An Outright EB is a complete EB on its own: it needs no `EBContext` and no leg linkage. To illustrate how an Outright EB is represented in both the J-encoding and F-encoding, we consider the following EB:

instrument	`91282CPJ4`
side	`buy`
maximum quantity	\$10000
limit	$P \leq 98.14453125$
lifetime	`3 auctions`

2.3.1. J-Encoding

Using JSON, we represent this EB as follows:

```
{
  "v1": {
    "outright": {
      "side": "buy",
      "max_qty": 10000,
      "limit": {
        "price_clean": 98.14453125
      },
      "instrument_id": "91282CPJ4",
      "in_force": {
        "for_auctions": 3
      },
      "id": "EB-002"
    }
  }
}
```

Once again, the J-encoding of this EB consists of just one message, a `NewOrderSingle` (`MsgType=D`) where the entire JSON string above is the value of FIX tag `20050` and no `ClOrdID` is present (`D18`):

Tag	Description	Value
	J-Encoding NewOrderSingle (`MsgType=D`)	
8	FIXVersion	FIX.4.4
35	MsgType	D
34	MessageSequenceNum	1240
49	SenderCompID	TRDR
52	SendTime	20260413-16:23:49.101
56	TargetCompID	OCMR
20050	EBJson	{"v1": {"outright": {"side": "buy", "max_qty": 10000, "limit": {"price_clean": 98.14453125}, "instrument_id": "91282CPJ4", "in_force": {"for_auctions": 3}, "id": "EB-002"}}}

2.3.2. F-Encoding

Since an Outright EB maps directly to a `NewOrderSingle` (`MsgType=D`) message, its corresponding F-encoding consists of a single FIX message with custom tags used to identify both the EB and the

© 2026 OneChronos Markets LLC

limit convention. That message carries the lifetime fields an `EBContext` carries for a MultiLeg EB, under the same rules: `TimeInForce` (59), `InForceAuctions` (20033), and `InForceSeconds` (20034), mutually exclusive and each at least `1`. The example's `in_force` of three auctions appears in the J-encoded payload above and as `InForceAuctions` here:

Tag	Description	Value
	F-Encoding NewOrderSingle (`MsgType=D`)	
8	FIXVersion	FIX.4.4
35	MsgType	D
34	MessageSequenceNum	1241
49	SenderCompID	TRDR
52	SendTime	20260413-16:23:49.202
56	TargetCompID	OCMR
11	ClOrdID	EB-002
20000	EBID	EB-002
48	SecurityID	91282CPJ4
22	SecurityIDSource	1
54	Side	1
38	OrderQty	10000
44	LimitPriceOrLimitYield	98.14453125
20031	LimitIsYield	N
20100	EBVersion	1
20033	InForceAuctions	3

2.4. JSON Schema

The TypeScript schema below makes the structure of an EB JSON payload explicit. It describes the value carried in FIX tag `20050` when using J-encoding:

```
export type ExpressiveBidJEncoding = {
  v1: Eb;
};
export type Eb =
  | {
      multi_leg: MultiLeg;
    }
  | {
      outright: Outright;
    };
export type InForce =
  | {
      for_time_seconds: number;
    }
  | {
      for_auctions: number;
    };
export type PartialFillBehavior =
  | "rest_with_initial_qty"
  | "cancel_on_fill"
  | "rest_without_initial_qty";
export type Side = "buy" | "sell";
export type Limit =
  | {
```

```
      price_clean: number;
    }
  | {
      yield: number;
    };

export interface MultiLeg {
  id: string;
  in_force?: InForce | null;
  instruments: {
    [k: string]: Leg;
  };
  min_qty?: MinQty | null;
  neutrality_tolerance?: number | null;
  partial_fill_behavior?: PartialFillBehavior | null;
  yield_constraints?: YieldConstraint[] | null;
}
export interface Leg {
  DV01?: number | null;
  initial_qty?: number | null;
  max_buy_qty?: number | null;
  max_sell_qty?: number | null;
}
export interface MinQty {
  first_only?: boolean | null;
  value: number;
}
export interface YieldConstraint {
  lower_bound: number;
  terms: {
    [k: string]: Term;
  };
}
export interface Term {
  alpha: number;
  side?: Side | null;
}
export interface Outright {
  dv01?: number | null;
  id: string;
  in_force?: InForce | null;
  instrument_id: string;
  limit?: Limit | null;
  max_qty: number;
  side: Side;
}
```

3. EB Lifecycle

Matching on OCM-R is organized as a sequence of discrete batch auctions. Each **auction cycle** in this sequence consists of two phases: an accumulation phase and a matching phase. New EBs can be submitted at any time and are eligible for matching in the auction that takes place after the EB is accepted, and every auction thereafter. In the 500 millisecond matching phase, the venue jointly evaluates all EBs for matching, and these EBs are **locked**.

A cancel request is woven into the auction cycle, not applied instantly. A cancel request against a resting EB is always accepted, and the cancellation eliminates the possibility of unfilled interest from being matched in subsequent auctions. If a cancel request is received during the matching phase (when EBs are locked), the EB may still result in a fill, either partial or full. Any residual EB interest is then canceled, and the EB will not participate in subsequent auctions. Cancellation is further described in section 5.2.

© 2026 OneChronos Markets LLC

The state of an EB is reflected to a subscriber through any number of `ExecutionReports` (`MsgType=8`) carrying the matching `OrdStatus`:

OrdStatus	State	Meaning
`A`	Pending New	Received and validating.
`0`	New	Accepted and resting on the OCM-R book.
`1`	Partially Filled	Some quantity filled, the remainder rests.
`6`	Pending Cancel	Cancel acknowledged, not yet applied.
`2`	Filled	Fully filled (**terminal**).
`4`	Canceled	Removed (**terminal**).
`8`	Rejected	Not accepted (**terminal**).

When a leg of an EB is filled, OCM-R reports the fill using an `ExecutionReport` message. Each filled leg of an EB is reported with a separate message. A partially filled leg carries `OrdStatus 1` (Partially Filled), distinct from the terminal `2` (Filled) that ends the life of an EB. Fill quantities are in increments of $1,000 of par notional value. Fill prices correspond to clean prices in increments of $\$\frac{1}{512}$.

For an EB constructed using F-encoding, a cancel request may arrive before the EB is constructed. A cancel request naming either a leg or the EB's own `EBID` cancels assembly of the partially constructed EB, and all the messages corresponding to that EB are removed.

4. EB Identifiers

An EB is named by its **EB identifier**. OCM-R echoes it on every `ExecutionReport` for the EB, and a cancel targets it. An EB is identified differently depending on the context:

- J-encoded EB: `id` field within EBJson (tag `20050`)
- F-encoded, MultiLeg EB: identified by EBID (tag `20000`)
- F-encoded, Outright EB: `ClOrdID` (tag `11`)

In an F-encoded MultiLeg EB, each leg message's `ClOrdID` (tag `11`) is referenced by the `EBContext` message's LegID (tag `20001`).

An `OrderCancelRequest` (`MsgType=F`) sets `OrigClOrdID` (tag `41`) to the EB identifier being canceled, while the request's own `ClOrdID` labels the cancel. An identifier is in use from acknowledgment until its EB is `Filled`, `Canceled`, or `Rejected`, and while in use it may name neither a new EB nor a leg of one (rule `C02`). After an EB ends, its identifier may be reused, though we recommend a fresh, session-unique identifier for every EB.

© 2026 OneChronos Markets LLC

5. FIX Specification

What follows is the FIX-based protocol for submitting and managing EBs. The protocol runs over a standard FIX 4.4 session. The application messages carry EB trading intent or report on EB state or state transitions.

5.1. Message Catalog

The two tables below list every message a subscriber may send to OCM-R and every message OCM-R will send to the subscriber. The session-level (admin) messages follow standard FIX 4.4 semantics; the detailed field-level specification for each application message follows.

5.1.1. Subscriber to OCM-R

Code	Message	Summary
`A`	Logon	Open an authenticated session.
`5`	Logout	Close the session.
`0`	Heartbeat	Keepalive. Also answers a Test Request.
`1`	Test Request	Ask OCM-R to prove liveness with a Heartbeat.
`2`	Resend Request	Ask OCM-R to replay messages missed after a sequence gap.
`4`	Sequence Reset	Advance the expected sequence number (gap fill).
`D`	New Order Single	Submit an Outright EB, or one leg of a MultiLeg EB.
`U9`	EBContext	Submit the binding/contextualization of legs in a MultiLeg EB.
`F`	Order Cancel Request	Cancel a resting or pending EB.

5.1.2. OCM-R to Subscriber

Code	Message	Summary
`A`	Logon	Accept the session.
`5`	Logout	Close the session.
`0`	Heartbeat	Keepalive. Also answers a Test Request.
`1`	Test Request	Ask the subscriber to prove liveness with a Heartbeat.
`2`	Resend Request	Ask the subscriber to replay messages missed after a gap.
`4`	Sequence Reset	Advance the expected sequence number (gap fill).
`3`	Reject	Session-level rejection of a malformed message.

© 2026 OneChronos Markets LLC

`8`	Execution Report	EB acknowledgments, fills, completion, cancel reports, and novation-risk heartbeats (`ExecType=I`).
`9`	Order Cancel Reject	Reject an Order Cancel Request. The EB is left untouched.
`j`	Business Message Reject	Application-level rejection of an unusable message.

© 2026 OneChronos Markets LLC

Identifiers

Every name on the wire is a short opaque string: an EB identifier, a message identifier, a session identifier. Two character sets apply, one per direction. The venue accepts a wider set than it mints, so an identifier you choose always comes back byte for byte, and an identifier the venue chooses is always safe to use as a map key, a filename, or a field in a delimited file.

Direction	Accepted characters	Length	Refused by
subscriber → OneChronos	printable ASCII `0x21` to `0x7E`, excluding `,` `;` `|`	1 to 32	A05 A06
OneChronos → subscriber	`a-z`, `A-Z`, `0-9`, and `# - . : _`	1 to 32	venue guarantee

The narrower outbound set covers the identifiers the venue mints for itself, `ExecID` above all. An identifier the venue is quoting back to you (`OrderID` and `ClOrdID` on an `ExecutionReport`, `OrigClOrdID` on an `OrderCancelReject`) is reproduced exactly as it arrived, so it stays in whichever of the two sets you chose it from. Space is excluded from both sets, so an identifier is always one whitespace-delimited token.

Tag	Name	Set by	Names
11	ClOrdID	subscriber	One F-encoded message: an outright or a leg. It is not the name of the EB (D18). On an `ExecutionReport` it carries the EB the report is about.
41	OrigClOrdID	subscriber	The EB an `OrderCancelRequest` targets, by its `EBID` (C28).
17	ExecID	venue	One `ExecutionReport`. Unique for the life of the session.
37	OrderID	venue	The EB a report is about, by its `EBID`. A report tied to no single EB, such as the novation-risk heartbeat, carries `NONE` rather than omitting the tag.
49	SenderCompID	both	The sending side of the session, from onboarding.
56	TargetCompID	both	The receiving side of the session, from onboarding.
112	TestReqID	subscriber	One `TestRequest`. The answering `Heartbeat` echoes it verbatim.
20000	EBID	subscriber	The EB itself, across every message and report that touches it.
20001	LegID	subscriber	A leg of an F-encoded basket, by repeating that leg message's `ClOrdID`.

Choosing a ClOrdID

A `ClOrdID` must be unique among the messages you currently have live, and an `EBID` unique among the EBs you currently have live. Uniqueness is checked only against that live set: an identifier whose message never assembled, or whose EB has gone terminal, is no longer held against you,

© 2026 OneChronos Markets LLC

and the venue will accept it again. Reusing one is still a poor idea, because your own reports for the two are then indistinguishable.

The venue does not impose a scheme beyond the character set. A workable one is to prepend the trade date and the session identifier to a per-session counter, for example `20260829:TRDR:000042`, which is unique by construction, sorts chronologically, and stays inside 32 characters. Identifiers are compared as bytes, so `EB-1` and `eb-1` are two different EBs.

© 2026 OneChronos Markets LLC

Messages

In every message table below, a ✓ in the Required column marks a field that must be present on the message: omitting it is a rejection, not a default. A field with no mark is optional, and the message means the same thing whether or not you send it. Required and optional describe this venue's contract, not the base FIX 4.4 dictionary, which is looser in both directions. Tables for messages the venue sends have no Required column: everything the venue promises to populate is described in the notes beneath the table.

Logon (`MsgType=A`)

Opens the FIX session. `EncryptMethod` (`98`) is required by FIX, but the venue does no message-level encryption, so it is always `0` (None). `HeartBtInt` (`108`) is the session heartbeat interval in seconds. The authentication token, formed from the Login Secret in your onboarding email, goes in `RawData` (`96`), with its byte length in `RawDataLength` (`95`). A `Logon` whose token does not verify is rejected.

Tag	Name	Required	FIX Type
			FIX Message: Logon (`35=A`)
98	EncryptMethod	✓	INT
108	HeartBtInt	✓	INT
141	ResetSeqNumFlag		BOOLEAN
95	RawDataLength	✓	INT
96	RawData	✓	DATA

Outright (`MsgType=D`) (J-Encoding)

A submission in the J-encoding: a single `NewOrderSingle` carrying the entire versioned EB, Outright or `MultiLeg`, as the payload of `EBJson` (20050). Three message shapes share `MsgType` D; a J-encoded EB is the one carrying `EBJson`. The EB's ID is carried inside the JSON payload, rather than as a value of tag `ClOrdID` (D18).

Tag	Name	Required	FIX Type
			FIX Message: Outright (J-Encoding) (`35=D`)
20050	EBJson	✓	STRING

VALIDATION RULES D18 D19

Outright (MsgType=D) (F-Encoding)

An Outright EB submission in the F-encoding: interest to trade a single instrument with a size and a limit (which could represent a clean price or a yield), spread across FIX tags. Three message shapes share MsgType D; an F-encoded Outright is the one carrying EBID (20000) without EBJson (20050). EBID names the EB, exactly as it does on an EBContext; ClOrdID names this message, and is the identifier echoed back on the reports it generates. DV01 (20005) is optional and overrides the instrument universe's value, matching the optional dv01 an Outright carries in the J-encoding.

Tag	Name	Required	FIX Type
		FIX Message: Outright (F-Encoding) (35=D)	
11	ClOrdID	✓	STRING
20000	EBID	✓	STRING
48	SecurityID	✓	STRING
22	SecurityIDSource	✓	SecurityIDSource
54	Side	✓	Side
38	OrderQty	✓	INT
44	LimitPriceOrLimitYield	✓	FLOAT
20031	LimitIsYield		BOOLEAN
20005	DV01		FLOAT
59	TimeInForce		TimeInForce
20033	InForceAuctions		INT
20034	InForceSeconds		INT
20100	EBVersion	✓	INT

VALIDATION RULES D04 D05 D06 D07 D08 D09 D10 D11 D17 D19

Leg (MsgType=D) (F-Encoding)

A leg of an EB whose structure is supplied by a separate EBContext (U9) message. EBContext references a leg using the leg's ClOrdID. A leg includes its quantity and side, but not its limit price (or limit yield). A leg does not repeat EBVersion: the version is specified once, as part of the EBContext message. Legs exist only in the F-encoding; a J-encoded EB carries its legs inside the EBJson payload.

Tag	Name	Required	FIX Type
			FIX Message: Leg (35=D)
11	ClOrdID	✓	STRING

© 2026 OneChronos Markets LLC

48	SecurityID	✓	STRING
22	SecurityIDSource	✓	SecurityIDSource
54	Side		Side
38	OrderQty		INT
20004	OtherOrderQty		INT

VALIDATION RULES D12 D05 D07 D13 D16 D19

EBContext (MsgType=U9) (F-Encoding)

Supplies the structure of an EB: its legs, neutrality tolerance, yield constraints, minimum quantity, partial-fill behavior, and lifetime. An EBContext references its legs through their ClOrdID values. EBContext exists only in the F-encoding; in the J-encoding, the same EB structure is carried as the payload of EBJson (20050). An EBContext identifies itself with EBID (20000) and does not carry a ClOrdID of its own (D18); the ClOrdIDs it references are those of its legs (through LegID).

Tag	Name	Required	FIX Type
			FIX Message: EBContext (35=U9)
20100	EBVersion	✓	INT
20000	EBID	✓	STRING
20010	NumLegs **(FIX Group)**	✓	NUMINGROUP
20001	LegID	✓	STRING
20005	DV01		FLOAT
20006	InitialQty		FLOAT
20012	NeutralityTolerance		FLOAT
20013	MinQty		INT
20014	MinQtyFirstOnly		BOOLEAN
20020	NumYieldConstraints **(FIX Group)**		NUMINGROUP
20021	ConstraintID	✓	STRING
20022	LowerBound	✓	FLOAT
20023	NumConstraintTerms **(FIX Group)**		NUMINGROUP
20024	TermSecurityID	✓	STRING
20025	TermAlpha	✓	FLOAT
20026	TermSide		TermSide
20030	PartialFillBehavior		PartialFillBehavior

© 2026 OneChronos Markets LLC

59	TimeInForce	TimeInForce
20033	InForceAuctions	INT
20034	InForceSeconds	INT

VALIDATION RULES D14 D06 D15 D09 D10 D11 D17 D18 D19 D20

ExecutionReport (MsgType=8)

Confirms receipt of an EB, relays EB status and fill information, and reports rejections. Sent by the venue to the subscriber's FIX session. ExecType identifies the report kind; ExecType=RiskHeartbeat marks a novation-risk snapshot rather than a fill.

Tag	Name	FIX Type
		FIX Message: ExecutionReport (35=8)
37	OrderID	STRING
11	ClOrdID	STRING
17	ExecID	STRING
150	ExecType	ExecType
39	OrdStatus	OrdStatus
48	SecurityID	STRING
22	SecurityIDSource	SecurityIDSource
54	Side	Side
32	LastQty	INT
31	LastPx	FLOAT
151	LeavesQty	INT
14	CumQty	INT
6	AvgPx	FLOAT
20060	Yield	FLOAT
20110	Risk	FLOAT
20111	RiskInBreach	BOOLEAN
20112	RiskLimit	FLOAT
20113	RiskRatio	FLOAT
58	Text	STRING

When ExecType is RiskHeartbeat, the venue guarantees that Risk, RiskInBreach, RiskLimit, and RiskRatio are present. On a rejection, Text carries the human-readable reason from the Error Message Catalog.

© 2026 OneChronos Markets LLC

BusinessMessageReject (MsgType=j)

Rejects an application message that cannot be processed, carrying the reason and the offending reference.

Tag	Name	FIX Type
		FIX Message: BusinessMessageReject (35=j)
45	RefSeqNum	INT
372	RefMsgType	STRING
379	BusinessRejectRefID	STRING
380	BusinessRejectReason	INT
58	Text	STRING

© 2026 OneChronos Markets LLC

Fields

`OrderQty` (38), `OtherOrderQty` (20004), `LastQty` (32), `LeavesQty` (151), and `CumQty` (14) are integer amounts in U.S. dollars. `InitialQty` (20006) is a signed floating-point amount in U.S. dollars; an explicit zero denotes a zero initial position.

The standard FIX fields used across these messages. The OneChronos 20000-block extensions have their own Custom tags section below.

Tag	Name	FIX Type
6	AvgPx	FLOAT
11	ClOrdID	STRING
14	CumQty	INT
17	ExecID	STRING
22	SecurityIDSource	SecurityIDSource
31	LastPx	FLOAT
32	LastQty	INT
37	OrderID	STRING
38	OrderQty	INT
39	OrdStatus	OrdStatus
44	LimitPriceOrLimitYield	FLOAT
45	RefSeqNum	INT
48	SecurityID	STRING
54	Side	Side
58	Text	STRING
59	TimeInForce	TimeInForce
150	ExecType	ExecType
151	LeavesQty	INT
372	RefMsgType	STRING
379	BusinessRejectRefID	STRING
380	BusinessRejectReason	INT

Custom Tags

OneChronos Expressive Bidding extensions in the 20000 block.

© 2026 OneChronos Markets LLC

Tag	Name	Purpose
20000	EBID	EB identifier
20001	LegID	Leg identifier within a basket
20004	OtherOrderQty	Opposing-side quantity for a two-sided leg
20005	DV01	DV01 associated with a leg's instrument.
20006	InitialQty	Signed initial quantity for a leg; an explicit zero denotes a zero initial position.
20010	NumLegs	count of legs in a basket.
20012	NeutralityTolerance	Upper bound on tolerated non-neutrality; absent when the EB carries no neutrality constraint.
20013	MinQty	Optional EB-level minimum aggregate filled par-notional. If the EB is filled in an auction, the basket must meet this bound, or it receives nothing.
20014	MinQtyFirstOnly	When Y (the default if MinQty is present and this tag is omitted), minimum quantity applies only to the first auction that fills the EB. When N, the bound applies in every filling auction. Meaningful only with MinQty (D20).
20020	NumYieldConstraints	count of yield constraints.
20021	ConstraintID	Yield constraint identifier
20022	LowerBound	Yield constraint lower bound
20023	NumConstraintTerms	Term count within a yield constraint
20024	TermSecurityID	Instrument the constraint term references
20025	TermAlpha	Yield coefficient for a term
20026	TermSide	For a double-sided leg, the side to which the term applies
20030	PartialFillBehavior	Resting-fill behavior selector
20031	LimitIsYield	Whether the F-encoded limit is a yield.
20033	InForceAuctions	Exact auction-count lifetime of at least one auction
20034	InForceSeconds	Exact time lifetime in whole seconds, measured from acknowledgment
20050	EBJson	Entire (versioned) EB as one JSON string (J-encoding)
20060	Yield	Yield reported on fills
20100	EBVersion	EB schema revision, carried once on the F-encoded EBContext message
20110	Risk	Current novation-risk value
20111	RiskInBreach	Whether current novation risk is in breach
20112	RiskLimit	Novation-risk limit
20113	RiskRatio	Quotient of Risk to RiskLimit

© 2026 OneChronos Markets LLC

Enumeration Types

Side

Encoding: `char` `Side` of a submission

Case	Tag
Buy	1
Sell	2
Both	U

OrdStatus

Encoding: `char` Current status of a submission

Case	Tag
New	0
PartiallyFilled	1
Filled	2
Canceled	4
PendingCancel	6
Rejected	8
PendingNew	A

ExecType

Encoding: `char` Describes the specific `ExecutionReport`

Case	Tag
New	0
PartialFill	1
Fill	2
Canceled	4
PendingCancel	6
Rejected	8
PendingNew	A
RiskHeartbeat	I

© 2026 OneChronos Markets LLC

SecurityIDSource

Encoding: `string` Class or source of the `SecurityID` value

Case	Tag
CUSIP	1

TermSide

Encoding: `char` `Side` of a constraint term

Case	Tag
Buy	1
Sell	2

TimeInForce

Encoding: `char` Lifetime of an EB

Case	Tag
Day	0
ImmediateOrCancel	3

PartialFillBehavior

Encoding: `int` How a partially filled resting bid behaves

Case	Tag
CancelOnFill	0
RestWithoutInitialQty	1
RestWithInitialQty	2

© 2026 OneChronos Markets LLC

Expressive Bid Validation

An EB is accepted only when every rule below that applies to it passes. The rules run in the order shown, top to bottom. A scope heading names the part of the EB being checked and says whether the rules run once or once per member of a collection. Each rule is stated in plain text and in formal form below. For example `C04`:

Code	Requirement	Applied over FIX Fields

`C04`	**Neutrality Tolerance Is in Range**
	When supplied, neutrality tolerance must be between 0.000001 and 1 inclusive.
	`when eb.neutralityTolerance is present require`
	`eb.neutralityTolerance ≥ 0.000001`
	`and`
	`eb.neutralityTolerance ≤ 1`
		J:20050, F:20012

Code is the rejection code returned when the rule fails. **Requirement** is the condition that must hold for the EB to remain valid. The **Applied over FIX Fields** column identifies the corresponding FIX fields. A reference is written as an encoding followed by a FIX tag: `J:20050` means the value is represented within the JSON EB payload carried in FIX tag 20050, while `F:20012` means it is represented directly by the F-encoded FIX field at tag 20012. Multiple references show where the same semantic input appears across the supported encodings.

The rules run in one canonical order, including collection traversal as displayed. A submission may violate several rules at once; the venue reports only the first failure. That failure surfaces on the subscriber's FIX session in the rejection message associated with the request, and its code and description appear in the Error Message Catalog. Decode and assembly failures that occur before these semantic rules are also listed in that catalog.

SubmitExpressiveBid

Whole EB once per EB

Code	Requirement	Applied over FIX Fields
`C01`	**EB Identifier Is Present** The EB identifier must not be empty. `require stringLength ( eb.id ) > 0`	J:20050, F:20000
`C02`	**EB Identifier Is Unique** The EB identifier must not already be in use by the subscriber, as a live EB or as a pending F-encoded leg or basket. `require`	J:20050, F:20000

© 2026 OneChronos Markets LLC

```
not
  eb.id belongs to state.subscriberEbIds
```

C03 **EB Has at Least One Leg** J:20050, F:20010

The EB must contain at least one leg.

```
require count ( eb.legs ) > 0
```

C04 **Neutrality Tolerance Is in Range** J:20050, F:20012

When supplied, neutrality tolerance must be between 0.000001 and 1 inclusive.

```
when eb.neutralityTolerance is present require
  eb.neutralityTolerance ≥ 0.000001
  and
  eb.neutralityTolerance ≤ 1
```

C05 **Neutrality Uses a Non-Zero DV01** J:20050, F:20010, F:20012, F:20005

When neutrality tolerance is supplied, at least one leg must have a non-zero `DV01`.

```
when eb.neutralityTolerance is present require
  there is at least one leg in eb.legs where
    leg.DV01 is present
    and
    leg.DV01 ≠ 0
```

C06 **Time in Force Bound Is Positive** J:20050, F:20034, F:20033

Any supplied time-in-force bound must be positive.

```
require
  when
    eb.inForce.forTimeSeconds is present
  then
    eb.inForce.forTimeSeconds > 0
and
  when
    eb.inForce.forAuctions is present
  then
    eb.inForce.forAuctions > 0
```

C07 **Time in Force Matches Fill Behavior** J:20050, F:20033, F:20030

When partial-fill behavior is `CancelOnFill` or `RestWithoutInitialQty`, an auction-count bound must not exceed one.

```
when
  eb.inForce.forAuctions is present
  and
  eb.partialFillBehavior must be one of [ CancelOnFill
  RestWithoutInitialQty ]
require eb.inForce.forAuctions ≤ 1
```

C08 **Minimum Quantity Is Positive** J:20050, F:20013

When supplied, minimum quantity must be greater than zero.

© 2026 OneChronos Markets LLC

```
when eb.minQty is present require eb.minQty.value > 0
```

C09	**Minimum Quantity Is Within Tradeable Size** When supplied, minimum quantity must not exceed the sum of each leg's max(buy, sell). A leg fills in only one direction. `when eb.minQty is present require eb.minQty.value ≤ eb.tradeableQty`	J:20050, F:20013

Leg once per leg, ordered by instrument

Code	Requirement	Applied over FIX Fields
C10	**Leg Instrument Is Tradeable** Each leg instrument must be in the venue's tradeable universe. `require leg.instrument belongs to venue.tradeableInstruments`	J:20050, F:48
C11	**Leg CUSIP Is Well Formed** Each leg instrument must be a well-formed CUSIP. `require wellFormedCusip ( leg.instrument )`	J:20050, F:48
C12	**Leg CUSIP Check Digit Is Valid** Each CUSIP leg must have a valid check digit. `require validCusipChecksum ( leg.instrument )`	J:20050, F:48

C13	**Leg Quantity Limits Are Positive**
	Each supplied buy or sell quantity limit must be positive. An explicit `InitialQty` of zero is not an error; it denotes a zero initial position and counts as a supplied parameter.
	`require`
	`  when`
	`  leg.maxBuyQty is present`
	`  then`
	`  leg.maxBuyQty > 0`
	`and`
	`  when`
	`  leg.maxSellQty is present`
	`  then`
	`  leg.maxSellQty > 0`
		J:20050, F:38, F:20004

C14	**Leg Has at Least One Parameter**
	Each leg must supply at least one of `MaxBuyQty`, `MaxSellQty`, `InitialQty`, or `DV01`.
	`require`
	`leg.maxBuyQty is present`
	`or`
	`leg.maxSellQty is present`
	`or`
	`leg.initialQty is present`
	`or`
	`leg.DV01 is present`
		J:20050, F:20005, F:38, F:20004, F:20006

© 2026 OneChronos Markets LLC

C15 **Neutrality Legs Have DV01** J:20050, F:20012, F:20005

When neutrality tolerance is supplied, every leg must supply DV01.

```
when eb.neutralityTolerance is present require leg.DV01 is
present
```

C16 **Leg DV01 Is Finite** J:20050, F:20005

Every supplied leg DV01 must be finite.

```
when leg.DV01 is present require finite ( leg.DV01 )
```

C17 **Leg Quantities Fit the Supported Range** J:20050, F:38, F:20004, F:20006

Every supplied leg quantity must fit the venue's supported integer range.

```
require
  not
    exceedsMaxInteger ( leg.maxBuyQty , leg.maxSellQty ,
    leg.initialQty )
```

C18 **Leg Quantities Are Within the Venue Limit** J:20050, F:38, F:20004

Buy and sell quantities must not exceed the venue's maximum leg quantity.

```
require
  when
    leg.maxBuyQty is present
  then
    leg.maxBuyQty ≤ venue.maxOrderQty
and
  when
    leg.maxSellQty is present
  then
    leg.maxSellQty ≤ venue.maxOrderQty
```

C19 **Leg DV01 Is Within the Venue Band** J:20050, F:20005

Every supplied DV01 must be within the venue's configured DV01 band.

```
when leg.DV01 is present require
  when
    venue.DV01Min is present
  then
    leg.DV01 ≥ venue.DV01Min
and
  when
    venue.DV01Max is present
  then
    leg.DV01 ≤ venue.DV01Max
```

C20 **Leg DV01 Notional Is Within the Venue Limit**

A leg's maximum quantity multiplied by its DV01 must not exceed the configured cap.

© 2026 OneChronos Markets LLC

```
when venue.DV01NotionalMax is present require dv01Notional
( leg ) ≤ venue.DV01NotionalMax
```

Yield Constraint once per yield constraint

Code	Requirement	Applied over FIX Fields
C21	**Yield Bound Is Finite** Every yield-constraint lower bound must be finite. `require finite ( yield.lowerBound )`	J:20050, F:20022

C22	**Yield Bound Is Within the Venue Band**
	Every yield-constraint lower bound must be within the configured yield band.
	`require`
	`when venue.yieldMin is present then yield.lowerBound ≥ venue.yieldMin`
	`and`
	`when venue.yieldMax is present then yield.lowerBound ≤ venue.yieldMax`
		J:20050, F:20022

Yield Term once per term, ordered by instrument

Code	Requirement	Applied over FIX Fields
C23	**Term References an EB Leg** Each yield-constraint term instrument must identify a leg in the same EB. `require term.instrument belongs to eb.legs`	J:20050, F:20010, F:20024
C24	**Term Alpha Is Finite** Every yield-term coefficient must be finite. `require finite ( term.alpha )`	J:20050, F:20025
C25	**Term Alpha Is Non-Zero** Every yield-term coefficient must be non-zero. `require term.alpha ≠ 0`	J:20050, F:20025

C26	**Term Side Matches Leg Sidedness**
	`TermSide` must be present exactly when the referenced leg has both buy and sell quantities.
	`require`
	`term.side is present`
	`if and only if`
	`leg.maxBuyQty is present`
	`and`
	`leg.maxSellQty is present`
		J:20050, F:38, F:20004, F:20026

© 2026 OneChronos Markets LLC

MultiLeg EB once after all legs are available

Code	Requirement	Applied over FIX Fields
C27	**EB Leg Count Is Within the Maximum** A `MultiLeg` EB must not contain more than 1024 legs. `require count ( eb.legs ) ≤ 1024`	J:20050, F:20010

CancelExpressiveBid

Cancellation Request once per request

Code	Requirement	Applied over FIX Fields

C28	**Cancel Target Exists**
	`OrigClOrdID` must name a live EB of the requesting subscriber by `EBID`, never a leg `ClOrdID`.
	`require`
	`there is at least one target in state.expressiveBids where`
	`target.ebId = message.origClOrdId`
	`and`
	`target.subscriber = session.subscriber`
		F:41

C29	**Cancel Target Is Not Fully Filled**
	A cancel held through a matching phase is rejected when that phase filled the target EB completely.
	`require`
	`there is at least one target in state.expressiveBids where`
	`target.ebId = message.origClOrdId`
	`and`
	`not`
	`target.ordStatus is Filled`
		F:41

© 2026 OneChronos Markets LLC

Error Message Catalog

The catalog follows the order in which the venue processes a submission. A message first follows the decode path for its encoding: J-encoded messages pass through J-decoding, while F-encoded messages pass through F-decoding and then assembly. Once the venue has one complete EB, it applies the EB validation rules.

The rules form one canonical order. A message may have several problems at once; the venue reports only the first failure in that order. The returned code identifies that first failing rule on the applicable path, and every rejected message maps to exactly one code below.

J-Encoding Decode

Tag 20050 did not parse into a versioned EB.

Code	Response	Description
J01	BusinessReject	the `EBJson` (20050) value is not parseable JSON
J02	BusinessReject	the top-level version key is not a supported EB version (only v1 is defined)
J03	BusinessReject	the message carries no `EBJson` (20050) value
J04	BusinessReject	the JSON payload does not include a version (expected { "v1": ... })

F-Encoding Decode

A field was malformed or a message-shape rule failed.

Code	Response	Description
D01	BusinessReject	`LimitIsYield` (20031) is not a recognized boolean value (Y or N)
D02	BusinessReject	`LimitIsYield` (20031) was supplied without a limit price (44)
D03	BusinessReject	`Side` (54) is not a recognized value

© 2026 OneChronos Markets LLC

Code	Rule	Messages	Requirement and Formal Form

D04	**Outright Required Fields**	Outright (F-Encoding)	An Outright EB must include every field marked required in its message table.
			`message.ClOrdID is required`
			`and`
			`message.EBID is required`
			`and`
			`message.SecurityID is required`
			`and`
			`message.IDSource is required`
			`and`
			`message.Side is required`
			`and`
			`message.OrderQty is required`
			`and`
			`message.LimitPriceOrLimitYield is required`
			`and`
			`message.EBVersion is required`

D05	**Positive Submitted Quantity**	Outright (F-Encoding), Leg	`OrderQty` must be greater than zero. `message.OrderQty > 0`
D06	**Supported EB Version**	Outright (F-Encoding), EBContext	`EBVersion` must equal 1. `message.EBVersion = 1`
D07	**CUSIP Identifier Source**	Outright (F-Encoding), Leg	`IDSource` must identify the `SecurityID` as a CUSIP. `message.IDSource must be one of [ CUSIP ]`
D08	**Outright Side**	Outright (F-Encoding)	An Outright EB `Side` must be Buy or Sell. `message.Side must be one of [ Buy Sell ]`
D09	**Supported Time in Force**	Outright (F-Encoding), EBContext	`TimeInForce`, when present, must be Day or `ImmediateOrCancel`. `message.TimeInForce must be one of [ Day ImmediateOrCancel ]`

D10	**Time Lifetime**	Outright (F-Encoding), EBContext	`InForceSeconds`, when present, must be at least one.
			`when`
			`  message.InForceSeconds is present`
			`then`
			`  message.InForceSeconds ≥ 1`

D11	**Auction Count Lifetime**	Outright (F-Encoding), EBContext	`InForceAuctions`, when present, must be at least one. `InForceAuctions`=1 is equivalent to `TimeInForce`=`ImmediateOrCancel`.
			`when`
			`  message.InForceAuctions is present`
			`then`
			`  message.InForceAuctions ≥ 1`

© 2026 OneChronos Markets LLC

Code	Rule	Messages	Requirement and Formal Form

D12	**Leg Required Fields**	Leg	A leg must include every field marked required in its message table.
			`message.ClOrdID is required`
			`and`
			`message.SecurityID is required`
			`and`
			`message.IDSource is required`

D13	**Leg Side**	Leg	A leg `Side` must be Buy, Sell, or Both. `message.Side must be one of [ Buy Sell Both ]`

D14	**EB Context Required Fields**	EBContext	An `EBContext` must include every field marked required in its message table.
			`message.EBVersion is required`
			`and`
			`message.EBID is required`

D15	**Partial Fill Behavior**	EBContext	`PartialFillBehavior`, when present, must be a supported value. `message.PartialFillBehavior must be one of [ CancelOnFill RestWithoutInitialQty RestWithInitialQty ]`

D16	**Sided Exactly When Trading**	Leg	`Side` is present exactly when at least one of `OrderQty` or `OtherOrderQty` is present.
			`message.Side is present`
			`if and only if`
			`message.OrderQty is present`
			`or`
			`message.OtherOrderQty is present`

D17	**Exclusive Lifetime Bound**	Outright (F-Encoding), EBContext	At most one of `TimeInForce`, `InForceAuctions`, and `InForceSeconds` can be specified.
			`not`
			`message.TimeInForce is present`
			`and`
			`message.InForceAuctions is present`
			`and`
			`not`
			`message.TimeInForce is present`
			`and`
			`message.InForceSeconds is present`
			`and`
			`not`
			`message.InForceAuctions is present`
			`and`
			`message.InForceSeconds is present`

D18	**EB Identifier Is Never a ClOrdID**	Outright (J-Encoding), EBContext	`ClOrdID` must not be present. A J-encoded EB is named inside `EBJson`, an F-encoded basket by `EBID`.
			`not`
			`message.ClOrdID is present`

© 2026 OneChronos Markets LLC

Code	Rule	Messages	Requirement and Formal Form

D19	**Resend Flag Is Never Set**	Outright (J-Encoding), Outright (F-Encoding), Leg, EBContext	`PossResend`, when present, must be N. The venue does not replay application messages.
			when
			  message.PossResend is present
			then
			  message.PossResend = N

D20	**MinQtyFirstOnly Requires MinQty**	EBContext	`MinQtyFirstOnly` may be present only when `MinQty` is present.
			when
			  message.MinQtyFirstOnly is present
			then
			  message.MinQty is present

F-Encoding Assembly

The per-tag messages could not be assembled into one EB.

Code	Response	Description
A01	BusinessReject	a leg message repeats the `ClOrdID` of a leg already awaiting assembly
A02	BusinessReject	an `EBContext` repeats the `EBID` of a basket already awaiting assembly
A03	BusinessReject	the same instrument appears on more than one leg of the basket
A04	BusinessReject	`NumLegs` does not match the number of leg entries in the `EBContext`
A05	BusinessReject	an identifier (`SenderCompID`, `ClOrdID`, or `EBID`) is longer than 32 characters
A06	BusinessReject	an identifier (`SenderCompID`, `ClOrdID`, or `EBID`) is empty or uses a character outside the accepted set: printable ASCII excluding space, comma, semicolon, and vertical bar
A07	BusinessReject	the same instrument appears in more than one term of a single yield constraint
A08	BusinessReject	a leg message carries no `SecurityID` (48)
A09	BusinessReject	a leg quantity does not fit the supported integer range
A10	BusinessReject	the `EBContext` references a `LegID` for which no leg message exists
A11	ExecReject	the `EBContext` did not arrive within the 5 second assembly window
A12	ExecReject	basket bust on cancel
A13	CancelReject	sibling leg never arrived during cancellation

© 2026 OneChronos Markets LLC

EB Validation

These rules run over the complete assembled EB.

Code	Response	Description
C01	ExecReject	missing EB id
C02	ExecReject	EB id already in use
C03	ExecReject	EB has no legs
C04	ExecReject	neutrality tolerance must be between 0.000001 and 1
C05	ExecReject	neutrality tolerance requires at least one non-zero leg DV01
C06	ExecReject	time-in-force bound must be positive
C07	ExecReject	auction-count bound exceeds one (1) for a non-resting partial-fill behavior
C08	ExecReject	minimum quantity must be positive
C09	ExecReject	minimum quantity exceeds tradeable size
C10	ExecReject	instrument not in the tradeable universe
C11	ExecReject	not a well-formed CUSIP
C12	ExecReject	fails CUSIP check digit
C13	ExecReject	a supplied buy or sell quantity limit is not positive
C14	ExecReject	leg has no quantity, initial quantity, or DV01 parameter
C15	ExecReject	neutrality tolerance requires DV01 on every leg
C16	ExecReject	DV01 is not finite
C17	ExecReject	quantity integer overflow
C18	ExecReject	quantity exceeds the venue size limit
C19	ExecReject	DV01 is outside the venue band
C20	ExecReject	DV01 notional exceeds the venue cap
C21	ExecReject	yield bound is not finite
C22	ExecReject	yield bound is outside the venue band
C23	ExecReject	yield-constraint term references an instrument that is not a leg of this EB
C24	ExecReject	yield term coefficient not finite
C25	ExecReject	yield term coefficient must be non-zero
C26	ExecReject	term.side presence does not match leg sidedness
C27	ExecReject	basket has too many legs
C28	CancelReject	cancel target EB not found
C29	CancelReject	cancel target EB was fully filled

© 2026 OneChronos Markets LLC

5.2. EB Cancellation

To cancel a resting EB, a subscriber sends an `OrderCancelRequest` (`MsgType=F`) whose `OrigClOrdID` (tag `41`) is the EB identifier to cancel (see EB Identifiers). OCM-R does not support amend or replace operations; to change a resting EB, a subscriber will cancel it and submit a new EB under a new identifier. A request naming an identifier that does not correspond to a live EB owned by the requesting Subscriber is refused with `OrderCancelReject` (`MsgType=9`).

Handling of cancellation requests is woven into the auction cycle. Broadly, EB cancellation can be classified into two categories:

- **Subscriber-triggered**: Occurs when a Subscriber submits a cancellation request.
- **Venue-triggered**: Occurs when an EB's lifetime expires or its partial-fill behavior requires cancellation.

A **subscriber-triggered** cancel request results in a **two-step process**. OCM-R first acknowledges the request with an `ExecutionReport` carrying `ExecType=PendingCancel` (`6`), then confirms removal with a second `ExecutionReport` carrying `ExecType=Canceled` (`4`) and the cumulative filled quantity in `CumQty` (`14`).

For an EB locked in a matching phase, any cancel confirmation occurs at the end of the phase. The precise message confirmations depend on the execution status of the EB at the end of an auction cycle's matching phase.

- if the EB went unfilled, it is removed from the OCM-R book and canceled with `ExecType=Canceled`.
- if the EB is fully filled, it is removed from the OCM-R book. The cancel request is rejected with `OrderCancelReject` (`MsgType=9`).
- if the EB is partially filled, it is removed from the OCM-R book and canceled with `ExecType=Canceled`.

A **venue-triggered** cancellation can occur under the following conditions:

The venue cancels when	When it occurs
`InForceAuctions=n` have taken place	At the end of the `n`-th auction cycle's matching phase
`InForceSeconds=s` have elapsed	After `s` seconds, or following the matching phase should the expiry time fall within it.
`PartialFillBehavior=CancelOnFill` and matching results in a fill	At the end of the matching phase

The following is a request to cancel the Outright EB `EB-002` from the earlier example: `OrigClOrdID` carries that EB identifier (highlighted) and `ClOrdID` labels the cancel itself:

Tag	Description	Value
	OrderCancelRequest (`MsgType=F`)	

8	FIXVersion	FIX.4.4
35	MsgType	F
34	MessageSequenceNum	1241
49	SenderCompID	TRDR
52	SendTime	20260413-16:24:02.317
56	TargetCompID	OCMR
41	OrigClOrdID	EB-002
11	ClOrdID	cxl-EB-002
55	Symbol	91282CPJ4
54	Side	1
60	TransactTime	20260413-16:24:02.317

© 2026 OneChronos Markets LLC

5.3. Novation Risk Heartbeats

OCM-R publishes asynchronous novation-risk snapshots to each active FIX session. Clearing and novation progress independently of an individual EB submission or fill; the venue therefore sends these snapshots as independent status messages rather than as a synchronous response to an EB. Clients should retain the latest snapshot and continue to process EB acknowledgments, fills, and rejects independently. Each heartbeat reflects the venue's latest view of the Subscriber's novation risk when the message is sent. Novation risk may change between heartbeats.

Risk snapshots use an `ExecutionReport` (`MsgType=8`). They are emitted on login and every 30 seconds thereafter. The message carries no EB identifier: it is a session-level report from the venue to the subscriber. `ExecType=RiskHeartbeat` (`I`) identifies the report as a novation-risk snapshot, distinguishing it from a fill report.

The risk measurements in this message use FIX `FLOAT`. `Risk` is the current novation-risk value, `RiskLimit` is its limit, and `RiskRatio` is the quotient `Risk / RiskLimit`.

Tag	FIX Type	Field	Meaning
		Novation Risk Heartbeat (`MsgType=8`)	
35	`CHAR`	`MsgType`	`8` (`ExecutionReport`)
150	`ExecType`	`ExecType`	`I` (`RiskHeartbeat`: a novation-risk snapshot, not a fill report)
20110	`FLOAT`	`Risk`	current novation-risk value
20111	`BOOLEAN`	`RiskInBreach`	whether current risk is in breach
20112	`FLOAT`	`RiskLimit`	novation-risk limit
20113	`FLOAT`	`RiskRatio`	quotient `Risk` / `RiskLimit`

A heartbeat carrying a subscriber's current risk looks like:

Tag	Description	Value
	Novation Risk Heartbeat (`MsgType=8`)	
8	FIXVersion	FIX.4.4
35	MsgType	8
34	MessageSequenceNum	84
49	SenderCompID	OCMR
52	SendTime	20260413-16:23:16.008
56	TargetCompID	TRDR
150	ExecType	I
20110	Risk	837500.2
20111	RiskInBreach	N
20112	RiskLimit	1000000
20113	RiskRatio	0.8375

© 2026 OneChronos Markets LLC

© 2026 OneChronos Markets LLC

6. The CERT Self-Certification Environment

To assist subscribers in developing FIX clients that target OCM-R, we host a self-certification environment called "CERT." For each testing subscriber, CERT creates an isolated FIX gateway and auction context in which the subscriber can self-match. CERT also offers a web UI for setting session parameters, inspecting message traffic, and creating EBs. The UI shows live EBs, fills, and execution reports next to a running FIX message log, so a subscriber can monitor the actions of its FIX clients as they occur. The CERT UI can also trigger auctions that match the subscriber's resting EBs.

The subscriber's FIX client connects to the gateway as the FIX **initiator**, with OCM-R as the **acceptor**:

Aspect	Behavior
FIX Version	FIX 4.4.
`SenderCompID`	Chosen by the client. OCM-R accepts any value.
`TargetCompID`	Fixed to `OCMR-CERT`.
Logon	The first message is a `Logon` (`MsgType=A`) carrying the subscriber's login token in `RawData`.
Sequence numbers	Standard FIX inbound and outbound sequence numbers, persisted across reconnects. Gaps recover with `ResendRequest` (`2`) and `SequenceReset` (`4`).
Sequence reset	A `Logon` with `ResetSeqNumFlag` (`141=Y`) starts a fresh session with both sides reset to 1. The sequence numbers can also be reset from the CERT UI.
Availability	CERT runs at any time, independent of production market hours. The CERT UI starts and stops the gateway.
Clock	The trading day and venue clock are driven from the CERT UI, which can set and simulate any market hours and defaults to the SIFMA calendar.

The CERT UI is accessible using credentials also provided during onboarding. These credentials are separate from the FIX Login Secret.

6.1. FIX Sample Client

In addition to CERT, OneChronos publishes `ocmr`, a reference Python FIX client. It includes short sample scripts that connect to the OCM-R CERT environment, submit EBs, and read back acknowledgments, fills, and risk heartbeats. This section describes the installation of the client and the execution of code samples against a subscriber's CERT FIX gateway. It assumes only basic familiarity with Python and a terminal environment on any operating system.

© 2026 OneChronos Markets LLC

Executing the code samples requires a host whose public IP address we allowlisted during onboarding, and the connection details we provided during onboarding: the CERT host and port, the `SenderCompID`, and the session secret. Python need not be installed: the samples use `uv`, which downloads and manages the required Python (3.12) and all dependencies automatically.

6.2. Install uv

`uv` is a Python package and project manager. Install it once, then reopen the terminal.

On macOS and Linux, in a terminal:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

On Windows, in PowerShell:

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

Confirm it works:

```
uv --version
```

6.3. Download the Client

OneChronos publishes the FIX client in the public ocmr-client repository on GitHub. Each release carries two build artifacts:

- Python wheel (.whl), the `ocmr` library on its own, for use inside an existing Python project: `uv pip install ocmr_reference-0.1.0-py3-none-any.whl`
- Python sdist (.tar.gz), the full source distribution including the sample scripts. Use this one for the walkthrough below.

Download the sdist, then unpack it and enter the directory. Where `tar` is available, use:

```
tar -xzf ocmr_reference-0.1.0.tar.gz
cd ocmr_reference-0.1.0
```

6.4. Configure the Session

Every sample accepts the same options: `--host`, `--port`, `--sender`, and `--target`. Rather than typing them each time, set the connection once as environment variables using the values we provided during onboarding.

© 2026 OneChronos Markets LLC

On macOS and Linux:

```
export CERT_FIX_HOST=<CERT host from onboarding>
export CERT_FIX_PORT=<CERT port from onboarding>
export CERT_FIX_SECRET=<session secret from onboarding>
```

On Windows PowerShell:

```
$env:CERT_FIX_HOST = "<CERT host from onboarding>"
$env:CERT_FIX_PORT = "<CERT port from onboarding>"
$env:CERT_FIX_SECRET = "<session secret from onboarding>"
```

6.5. Run the Samples

The first sample fetches Python 3.12 and the dependencies automatically; nothing else needs to be installed. Start with the logon check, passing the registered `SenderCompID`:

```
$ uv run 1_logon.py --sender TRDR01
Successfully connected to OneChronos Rates CERT
Successfully received risk heartbeat
Login Complete
```

Once that works, run the remaining samples in order, the same way:

- `1_logon.py`
- `2_send.py`
- `3_reject.py`
- `4_cancel.py`
- `5_replay.py`
- `6_fills.py`
- `7_poll.py`
- `8_constraint_reject.py`
- `9_order_validation.py`
- `risk_listen.py`

Each sample is self-contained. However, several samples only produce fills against opposing interest. The CERT UI can generate a matching counter-EB for any resting EB with a single click. Some samples only complete once an auction runs, and CERT does not run auctions on a wall-clock automatically: auctions must be triggered from the CERT UI.

© 2026 OneChronos Markets LLC

7. Type Index

For clarity, we use the following typographic conventions:

- FIX standard types are rendered below with blue pastel pills: `FIX`.
- custom FIX datatypes use a distinct blue pastel: `FIX ENUM`.
- JSON payload types use green pastel pills: `JSON`.

Every custom FIX datatype pill links to its allowed values in the FIX Specification.

7.1. FIX Types

Type	Meaning	Examples
`STRING`	text value	`EBID`, `ClOrdID`
`INT`	whole-number value	leg quantity, `NumLegs`
`FLOAT`	floating-point value	`Risk`, `RiskLimit`, `RiskRatio`
`CHAR`	single-character code	`MsgType`
`BOOLEAN`	true/false value	`RiskInBreach`
`NUMINGROUP`	counted repeating group	`NumLegs`, `NumYieldConstraints`
`UTCTIMESTAMP`	UTC date and time	`ExpireTime`
`Side`	custom FIX datatype	EB side
`OrdStatus`	custom FIX datatype	EB lifecycle status
`ExecType`	custom FIX datatype	`ExecutionReport` event
`SecurityIDSource`	custom FIX datatype	`SecurityID` source
`PartialFillBehavior`	custom FIX datatype	post-fill resting behavior
`TermSide`	custom FIX datatype	yield-constraint term side
`TimeInForce`	standard FIX enumeration	EB lifetime

7.2. JSON Types

Type	Meaning	Examples
`string`	JSON text value	`id`, `instrument_id`
`number`	JSON numeric value	`DV01`, `lower_bound`

© 2026 OneChronos Markets LLC

`boolean`	JSON true/false value	`limit_is_yield`, `first_only`
`array`	ordered JSON values	`yield_constraints`
`object`	named JSON values	`instruments`, `terms`

8. Disclaimers

Certain information contained in these materials may relate to a proposed business, market structure, or trading venue that has not yet been formed and is subject to regulatory approval and other conditions. There can be no assurance that any such business, platform, or services will be launched in the form described, or at all.

These materials are not, and under no circumstances should be construed as, an offer, solicitation, or recommendation to buy or sell any security, financial instrument, or trading strategy, nor do they constitute investment, legal, tax, accounting, or regulatory advice.

These materials are intended solely for institutional and professional market participants and may not be reproduced, distributed, or transmitted without the prior written consent of OneChronos.

© 2026 OneChronos Markets LLC