LSEG Websocket API

LSEG WEBSOCKET API FOR PRICING STREAMING AND REAL-TIME SERVICES
PROTOCOL SPECIFICATION AND DEVELOPERS GUIDE

Document Version: 1.4
Date of issue: November 2024
Document ID: WSA100LI.240

LSEG DATA & ANALYTICS

Contents

1 Product Introduction 1
1.1 Overview 1
1.2 Requirements 1
1.3 Protocol: tr_json2 1
1.4 Ping and Pongs 2

2 Login 3
2.1 Definition of the Login Domain 3
2.2 Login Structure 3
2.3 Login Context 4
2.3.1 Login 4
2.3.2 Login Response 4
2.3.3 Close Login 5

3 Item Request 6
3.1 Definition of Item Request 6
3.2 Structure 6
3.3 Context 6
3.3.1 Login 6
3.3.2 Login Response 7
3.3.3 Item Request 7
3.3.4 Item Response 8
3.3.5 Item Update 9
3.3.6 Close Item 10
3.3.7 Close Login 10

4 Close 11
4.1 Definition of Close 11
4.2 Structure 11
4.3 Context 12
4.3.1 Login 12
4.3.2 Login Response 12
4.3.3 Item Request 13
4.3.4 Item Response 13
4.3.5 Item Update 15
4.3.6 Close Item 16
4.3.7 Close Login 16

5 Authentication 17
5.1 Authentication Feature Description 17
5.2 Authentication Structure 18
5.3 Context 19
5.3.1 Login 19
5.3.2 Login Response 20
5.3.3 Close Login 20
5.4 Automatic Context 21
5.4.1 Automatic Login Response 21
5.4.2 Close Automatic Login 21

6 Batch 22

6.1 Definition of Batch 22
6.2 Structure 22
6.3 Context 22
6.3.1 Login 22
6.3.2 Login Response 23
6.3.3 Item Request 23
6.3.4 Batch Response 24
6.3.5 Item Response: TRI.N 24
6.3.6 Item Response: IBM.N 26
6.3.7 Item Response: T.N 28
6.3.8 Item Update(s) 30
6.3.9 Batch Close 30
6.3.10 Batch Close Response 31
6.3.11 Close Login 31

7 Posting 32
7.1 Definition of Posting 32
7.2 Structure 32
7.3 Context 34
7.3.1 Login 34
7.3.2 Login Response 1 35
7.3.3 Item Request 1 35
7.3.4 Item Response 36
7.3.5 Item Request 2 38
7.3.6 Login Response 2 38
7.3.7 Item Request 3 39
7.3.8 Item Response 2 39
7.3.9 Item Update 40
7.3.10 Post 40
7.3.11 Ack 41
7.3.12 Close Item 41
7.3.13 Close Login 41

8 View 42
8.1 Definition of View 42
8.2 Structure 42
8.3 Context 42
8.3.1 Login 42
8.3.2 Login Response 43
8.3.3 View Item Request 43
8.3.4 Item Response 44
8.3.5 Item Update(s) 44
8.3.6 Close Item 45
8.3.7 Close Login 45

9 Examples 46
9.1 Language-Specific Examples 46
9.2 Delivery Platform Connectivity Examples 46

10 Primitive Types 47

11 Container: Elements 49

12 Container: Fields 50

13 Container: Json 51

14 Container: Map 52
14.1 Members 52
14.2 For Example 53

15 Container: Message 54
15.1 Request Message 54
15.2 Packed Request Messages 54
15.3 Post Message 55

16 Container: Opaque 56

17 Container: Series 57
17.1 Members 57
17.2 For Example 57

18 Container: Vector 58
18.1 Members 58
18.2 For Example 58

19 Container: Xml 60

20 Messages: Ack Message 61
20.1 Ack Message Description 61
20.2 Ack Message Structure 61

21 Close 64
21.1 Close Message Description 64
21.2 Close Message Structure 64

22 Error Message 66
22.1 Error Message Description 66
22.2 Error Message Structure 66
22.3 Error: Unexpected Token Type 67
22.4 Error: Unexpected Parameter 68
22.5 Error: Missing Key 69
22.6 Error: Unexpected Key 70
22.7 Error: Unexpected Field IDentifier 71
22.8 Error: Array Type Mismatch 73

23 Generic Message 75
23.1 Generic Message Description 75
23.2 Generic Message Structure 75

24 Ping and Pong Messages 77
24.1 Ping and Pong Message Descriptions 77
24.2 Message Structure 77

25 Post Message 78
25.1 Post Message Description 78
25.2 Post Message Structure 78

26 Refresh Message 81
26.1 Refresh Message Description 81
26.2 Refresh Message Structure 81

27 Request Message 86
27.1 Request Message Description 86
27.2 Request Message Structure 86

28 Status Message 90
28.1 Status Message Description 90
28.2 Status Message Structure 90

29 Update Message 94
29.1 Update Message Description 94
29.2 Update Message Structure 94

30 Domain Model Usage: Market Price Domain 97
30.1 Market Price Domain Overview 97
30.2 Market Price Domain Examples 97
30.2.1 Market Price Request Message Sent 97
30.2.2 Market Price Refresh Message Received 97
30.2.3 Market Price Update Message Received 104
30.3 Usage: Market Price Request Message 105
30.4 Usage: Market Price Refresh Message 106
30.5 Usage: Market Price Update Message 108
30.6 Usage: Market Price Status Message 109

31 Domain Model Usage: Market by Price Domain 111
31.1 Market by Price Domain Overview 111
31.2 Market by Price Domain Examples 111
31.2.1 Market by Price Request Message Sent 111
31.2.2 Market by Price Refresh Message Received 111
31.2.3 Market by Price Update Message Received 113
31.3 Usage: Market by Price Refresh Message 116
31.4 Usage: Market by Price Update Message 118
31.5 Usage: Market by Price Status Message 120

32 Domain Model Usage: Market by Order Domain 121
32.1 Market by Order Domain Overview 121
32.2 Market by Order Domain Examples 121
32.2.1 Market by Order Request Message Sent 121
32.2.2 Market by Order Refresh Message Received 121
32.2.3 Market by Order Update Message Received 124
32.3 Usage: Market by Order Request Message 125
32.4 Usage: Market by Order Refresh Message 127
32.5 Usage: Market by Order Update Message 129
32.6 Usage: Market by Order Status Message 131

33 Domain Model Usage: Market Maker Domain 132
33.1 Market Maker Domain Overview 132
33.2 Market Maker Domain Examples 132
33.2.1 Market Maker Request Message Sent 132
33.2.2 Market Maker Refresh Message Received 132
33.2.3 Market Maker Update Message Received 135

33.3 Usage: Market Maker Request Message 136
33.4 Usage: Market Maker Refresh Message 137
33.5 Usage: Market Maker Update Message 138
33.6 Usage: Market Maker Status Message 139

34 Domain Model Usage: Yield Curve Domain 141
34.1 Yield Curve Domain Overview 141
34.2 Yield Curve Domain Examples 141
34.2.1 Yield Curve Request Message Sent 141
34.2.2 Yield Curve Refresh Message Received 141
34.2.3 Yield Curve Update Message Received 144
34.3 Usage: Yield Curve Request Message 145
34.4 Usage: Yield Curve Refresh Message 146
34.5 Usage: Yield Curve Update Message 147
34.6 Usage: Yield Curve Status Message 149

35 Domain Model Usage: Symbol List Domain 150
35.1 Symbol List Domain Overview 150
35.2 Symbol List Domain Examples 150
35.2.1 Symbol List Request Message Sent 150
35.2.2 Symbol List Refresh Message Received 150
35.2.3 Symbol List Update Message Received 152
35.3 Usage: Symbol List Request Message 153
35.4 Usage: Symbol List Refresh Message 154
35.5 Usage: Symbol List Update Message 156
35.6 Usage: Symbol List Status Message 158

1 Product Introduction

1.1 Overview

The LSEG WebSocket API is an interface to create direct WebSocket access to any Open Message Model Content via an LSEG Real-Time Advanced Distribution Server. The API leverages standard JSON and WebSocket protocols to be easy to implement and understand.

LSEG Real-Time Distribution System features such as authentication and compression are supported and no client API is required.

The WebSocket API is easily extensible to scripting environments (e.g. Python, R, etc) as well as any language or environment that supports WebSockets and JSON (e.g. Ruby, CSharp, Java, etc).

Throughout the documentation, message attributes are listed alphabetically. However, this ordering is arbitrary, and you can include them in any order within the message.

Market Price Request

```
{
    "ID": 2,
    "Key": {
        "Name": "TRI.N"
    }
}
```

1.2 Requirements

- Access to an LSEG Real-Time Advanced Distribution Server
- Programming language or environment with WebSocket and JSON libraries

1.3 Protocol: tr_json2

The WebSocket API uses the **tr_json2** protocol, which is a text-based JSON protocol representing Open Message Model constructs and featuring human readability.

tr_json2 follows JSON standards and all keys and values of the protocol are case sensitive.

To initialize a WebSocket connection for using the WebSocket API, set the subprotocol as **tr_json2** through the WebSocket library of choice or ensure that the **Sec-WebSocket-Protocol** header value in the initial WebSocket connection is **tr_json2**.

WebSocket Library (Node.js)

```
...
_websocket = new WebSocket(WS_URL, "tr_json2");
...
```

HTTP Header

```
...
Sec-WebSocket-Protocol: tr_json2
...
```

1.4 Ping and Pongs

The Websocket API leverages JSON Ping and Pong messages between endpoints to monitor connection health. For further details, refer to the Ping and Pong Messages topic.

2 Login

2.1 Definition of the Login Domain

The **Login** domain is used to create a context within a system access point. You must use this special model to access all other domain models. A **Login** must be the very first request and be streaming so that user context is maintained.

Access points use special logic in handling Logins and utilize them to retrieve the user's permission information. The user's permissions profile is used to authorize all other domain model interactions for that user.

Regarding the Login Response, MaxMsgSize is the maximum supported message size as configured by the LSEG Real-Time Advanced Distribution Server. Any message greater in size is rejected and results in a disconnection.

2.2 Login Structure

ATTRIBUTE	TYPE	DEFINITION
Domain	string,int	The domain model represented by this message (e.g. Login, MarketPrice, Headline, etc.). Defaults to Market Price if absent.
ID	int	Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or capability requested.
└ Elements	object	An Element List describing additional attributes of the item stream.
└ ApplicationId	string	The ID of the application to which the connection is made.
└ Position	string	The IP address position of the application logging in.
└ Name	string,array(string)	Name(s) of the information requested.
Type	string,int	The message classification (e.g. Request, Response, Update, etc.). Defaults to Request if absent.

Table 1: Login Structure

2.3 Login Context

2.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"256",
            "Position":"127.0.0.1"
        },
    "Name":"user"
    }
}
```

2.3.2 Login Response

```
[
  {
    "Domain":"Login",
    "Elements":{
        "MaxMsgSize":61440,
        "PingTimeout":30,
  },
  "ID":1,
  "Key":{
    "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"256",
        "ApplicationName":"ADS",
        "Position":"127.0.0.1",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"user"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted
    },
    "Type":"Refresh"
  }
]
```

2.3.3 Close Login

```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

3 Item Request

3.1 Definition of Item Request

The term Market Price is used to denote an item which contains trades, indicative quotes and the inside top of book quotes. It includes the last traded price(s), best bid(s)/offer(s), related value data such as: names, codes, etc. and the related derived data such as: net change, pen, close, high(s), low(s), etc.

The current model for level 1 data forms the basis of the Market Price domain. It includes different asset classes including equities, fixed income, commodities, money, FX and contributed quote data.

3.2 Structure

ATTRIBUTE	TYPE	DEFINITION
ID	int,array(int)	Integer value(s) representing the event stream. It can also be used to match the request and responses.
Key	int,array(int)	The key representing the data content or capability requested.
└ Name	string,array(string)	Name(s) of the information requested.

Table 2: Item Request Structure

3.3 Context

3.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"256",
            "Position":"127.0.0.1"
        },
        "Name":"user"
    }
}
```

3.3.2 Login Response


```
[
  {
    "Domain":"Login",
    "Elements":{
        "MaxMsgSize":61440,
        "PingTimeout":30,
  },
  "ID":1,
  "Key":{
    "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"256",
        "ApplicationName":"ADS",
        "Position":"127.0.0.1",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"user"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted
    },
    "Type":"Refresh"
  }
]
```


3.3.3 Item Request


```
{
    "ID":2,
    "Key":{
        "Name":"TRI.N"
    }
}
```

3.3.4 Item Response


```
[
  {
    "Fields":{
      "ACVOL_1":8719016,
      "ADJUST_CLS":392.8,
      "ASK":9000,
      "ASKSIZE":1,
      "ASKXID":"BOS",
      "ASK_MMID1":"BOS",
      "BID":0.01,
      "BIDSIZE":1,
      "BIDXID":"BOS",
      "BID_MMID1":"BOS",
      "BID_NET_CH":null,
      "BID_TICK_1":"⇩",
      "BLKCOUNT":5,
      "BLKVOLUM":116195,
      "CLOSE_ASK":398.1,
      "CLOSE_BID":398.01,
      "CTS_QUAL":"   ",
      "CUM_EX_MKR":"   ",
      "CURRENCY":"USD",
      "EXCHTIM":"21:00:01",
      "EXDIVDATE":null,
      "GV1_FLAG":null,
      "GV1_TEXT":"-",
      "HIGH_1":398.85,
      "HSTCLBDDAT":null,
      "HSTCLSDATE":"2017-11-28",
      "HST_CLOSE":392.8,
      "HST_CLSBID":null,
      "INSCOND":"   ",
      "INSPRC":null,
      "INSVOL":null,
      "IRGCOND":"132",
      "IRGPRC":398.85,
      "IRGVOL":144,
      "IRGXID":"CIN",
      "LOW_1":394.11,
      "NETCHNG_1":5.35,
      "NUM_MOVES":16844,
      "OFFCL_CODE":null,
      "OFF_CD_IND":"CUS",
      "OPENEXID":"PSE",
      "OPEN_PRC":396.5,
      "OPN_NETCH":3.7,
      "PCTCHNG":1.36,
      "PRCTCK_1":"⇧",
      "PRC_QL2":"   ",
      "PRC_QL_CD":"   ",
      "PREF_DISP":2254,
      "PRNTBCK":968902,
      "PROD_PERM":6560,
      "PROV_SYMB":"GOOG",
      "QUOTIM":"19:01:55",
```

```
      "QUOTIM_MS":84932000,
      "RDNDISPLAY":66,
      "RDN_EXCHD2":"NMQ",
      "RDN_EXCHID":"    ",
      "RECORDTYPE":113,
      "SALTIM":"19:01:52",
      "SALTIM_MS":84514000,
      "SEQNUM":1266750,
      "TIMCOR":null,
      "TIMCOR_MS":137197144,
      "TRADE_DATE":"2017-11-29",
      "TRDPRC_1":398.15,
      "TRDTIM_MS":75601000,
      "TRDVOL_1":26506,
      "TRDXID_1":"NAS",
      "TRD_UNITS":"2DP ",
      "TURNOVER":392.8,
      "VOL_X_PRC1":397.9481
    },
    "ID":2,
    "Key":{
      "Name":"TRI.N",
      "Service":"DF_RMDS"
    },
    "QOS":{
      "Rate":"TickByTick",
      "Timeliness":"Realtime"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"All is well"
    },
    "Type":"Refresh"
  }
]
```


3.3.5 Item Update


```
[
    {
        "Fields":{
            "ASK":401.54,
            "ASKSIZE":10,
            "ASKXID":"NAS",
            "ASK_MMID1":"NAS",
            "BID":401.5,
            "BIDSIZE":18,
            "BIDXID":"NAS",
            "BID_MMID1":"NAS",
            "BID_NET_CH":3.49,
            "BID_TICK_1":"\u00fe",
            "GV1_TEXT":"-",
            "QUOTIM":"14:40:32:000:000:000",
```

```
            "QUOTIM":"14:40:32:000:000:000",
            "QUOTIM_MS":52832000
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "Type":"Update",
        "UpdateType":"Quote"
    }
]
```

3.3.6 Close Item

```
{
    "ID":2,
    "Type":"Close"
}
```

3.3.7 Close Login

```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

4 Close

4.1 Definition of Close

The Close message cancels an outstanding request or stops an existing event stream.

4.2 Structure

ATTRIBUTE	TYPE	DEFINITION
Domain	string,int	The domain model represented by this message. Defaults to Market Price if absent. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ID	int,array(int)	Integer value(s) representing the stream(s) to close.

Table 3: Close Structure

ATTRIBUTE	TYPE	DEFINITION
Type	string,int	The message classification. Set to Close for Close message. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 3: Close Structure

4.3 Context

4.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"256",
            "Position":"127.0.0.1"
        },
        "Name":"user"
    }
}
```

4.3.2 Login Response

```
[
    {
        "Domain":"Login",
        "Elements":{
        "MaxMsgSize":61440,
        "PingTimeout":30
        },
        "ID":1,
        "Key":{
            "Elements":{
                "AllowSuspectData":1,
                "ApplicationId":"256",
                "ApplicationName":"ADS",
                "Position":"127.0.0.1",
                "ProvidePermissionExpressions":1,
                "ProvidePermissionProfile":0,
                "SingleOpen":1,
                "SupportBatchRequests":7,
```

```
                "SupportEnhancedSymbolList":1,
                "SupportOMMPost":1,
                "SupportOptimizedPauseResume":1,
                "SupportPauseResume":1,
                "SupportStandby":0,
                "SupportViewRequests":1
            },
            "Name":"user"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"Login accepted by host."
        },
        "Type":"Refresh"
    }
]
```


4.3.3 Item Request


```
{
    "ID":2,
    "Key":{
        "Name":"TRI.N"
    }
}
```


4.3.4 Item Response


```
[
    {
        "Fields":{
            "ACVOL_1":8719016,
            "ADJUST_CLS":392.8,
            "ASK":9000,
            "ASKSIZE":1,
            "ASKXID":"BOS",
            "ASK_MMID1":"BOS",
            "BID":0.01,
            "BIDSIZE":1,
            "BIDXID":"BOS",
            "BID_MMID1":"BOS",
            "BID_NET_CH":null,
             "BID_TICK_1":"⇧",
            "BLKCOUNT":5,
            "BLKVOLUM":116195,
            "CLOSE_ASK":398.1,
            "CLOSE_BID":398.01,
```

```
            "CTS_QUAL":"   ",
            "CUM_EX_MKR":"   ",
            "CURRENCY":"USD",
            "EXCHTIM":"21:00:01",
            "EXDIVDATE":null,
            "GV1_FLAG":null,
            "GV1_TEXT":"-",
            "HIGH_1":398.85,
            "HSTCLBDDAT":null,
            "HSTCLSDATE":"2017-11-28",
            "HST_CLOSE":392.8,
            "HST_CLSBID":null,
            "INSCOND":"   ",
            "INSPRC":null,
            "INSVOL":null,
            "IRGCOND":"132",
            "IRGPRC":398.85,
            "IRGVOL":144,
            "IRGXID":"CIN",
            "LOW_1":394.11,
            "NETCHNG_1":5.35,
            "NUM_MOVES":16844,
            "OFFCL_CODE":null,
            "OFF_CD_IND":"CUS",
            "OPENEXID":"PSE",
            "OPEN_PRC":396.5,
            "OPN_NETCH":3.7,
            "PCTCHNG":1.36,
            "PRCTCK_1":"⇧",
            "PRC_QL2":"   ",
            "PRC_QL_CD":"   ",
            "PREF_DISP":2254,
            "PRNTBCK":968902,
            "PROD_PERM":6560,
            "PROV_SYMB":"GOOG",
            "QUOTIM":"19:01:55",
            "QUOTIM_MS":84932000,
            "RDNDISPLAY":66,
            "RDN_EXCHD2":"NMQ",
            "RDN_EXCHID":"   ",
            "RECORDTYPE":113,
            "SALTIM":"19:01:52",
            "SALTIM_MS":84514000,
            "SEQNUM":1266750,
            "TIMCOR":null,
            "TIMCOR_MS":137197144,
            "TRADE_DATE":"2017-11-29",
            "TRDPRC_1":398.15,
            "TRDTIM_MS":75601000,
            "TRDVOL_1":26506,
```

```
            "TRDXID_1":"NAS",
            "TRD_UNITS":"2DP ",
            "TURNOVER":392.8,
            "VOL_X_PRC1":397.9481
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "QOS":{
            "Rate":"TickByTick",
            "Timeliness":"Realtime"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"All is well"
        },
        "Type":"Refresh"
    }
]
```


4.3.5 Item Update


```
[
    {
        "Fields":{
            "ASK":401.54,
            "ASKSIZE":10,
            "ASKXID":"NAS",
            "ASK_MMID1":"NAS",
            "BID":401.5,
            "BIDSIZE":18,
            "BIDXID":"NAS",
            "BID_MMID1":"NAS",
            "BID_NET_CH":3.49,
            "BID_TICK_1":"\u00fe",
            "GV1_TEXT":"-",
            "QUOTIM":"14:40:32:000:000:000",
            "QUOTIM_MS":52832000
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "Type":"Update",
```

```
        "UpdateType":"Quote"
    }
]
```

4.3.6 Close Item

```
{
    "ID":2,
    "Type":"Close"
}
```

4.3.7 Close Login

```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

5 Authentication

5.1 Authentication Feature Description

The WebSocket API is fully compatible with Delivery Platform Authentication.

Authentication can be done within a Login Request Message that contains token information. The user will receive either a Login Refresh Message if the Login was successful or a Status Message in the event of a failure.

An application can also perform an automatic Login by passing token credentials during the initial WebSocket connection to the LSEG Real-Time Advanced Distribution Server via HTTP Cookies. The names of these Cookies may be configured on the LSEG Real-Time Advanced Distribution Server using the following configuration parameters:

```
*ads*authTokenName : AuthToken
*ads*positionName : position
*ads*applicationIdName : applicationId
```

If the LSEG Real-Time Advanced Distribution Server detects the presence of Authentication credentials passed in HTTP Cookies during the initial WebSocket connection, the LSEG Real-Time Advanced Distribution Server will generate a Login Request Message for the user using the credentials passed. In this automatic Login case, the ID assigned to the generated Login Request will be a negative number, normally -1, as it is source generated.

5.2 Authentication Structure

ATTRIBUTE	TYPE	DEFINITION
Domain	string,int	The domain model represented by this message. Defaults to Market Price if absent. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ID	int	Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or capability requested.
└ Elements	object	An Element List describing additional attributes of the item stream.
└ ApplicationId	string	The ID of the application being connected to
└ AuthenticationToken	string	Login user authentication token
└ Position	string	The IP address position of the application logging in

Table 4: Authentication Structure

ATTRIBUTE	TYPE	DEFINITION
└ NameType	string,int	An enumeration representing the different forms the name can take. • AuthnToken: Authentication Token. • Cookie: User information is specified in cookie. • EmailAddress: Email Address. • Name: Username. • Ric: Reuters Instrument Code. • Token: User Token (Typically AAA Token). • Unspecified: Unspecified.

Table 4: Authentication Structure

5.3 Context

5.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"555",
            "AuthenticationToken":"aBcDeFgHiJkLmNoPqRsTuVwXyZ",
            "Position":"127.0.0.1"
        },
        "NameType":"AuthnToken"
    }
}
```

5.3.2 Login Response


```
[
  {
    "Domain":"Login",
    "Elements":{
      "MaxMsgSize":61440,
      "PingTimeout":30
    },
    "ID":1,
    "Key":{
      "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"555",
        "ApplicationName":"ADS",
        "AuthenticationErrorCode":0,
        "AuthenticationErrorText":"Success",
        "Position":"127.0.0.1",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"user"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted by host."
    },
    "Type":"Refresh"
  }
]
```


5.3.3 Close Login


```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

5.4 Automatic Context

5.4.1 Automatic Login Response


```
[
  {
    "Domain":"Login",
    "Elements":{
      "MaxMsgSize":61440,
      "PingTimeout":30
    },
    "ID":-1,
    "Key":{
      "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"555",
        "ApplicationName":"ADS",
        "AuthenticationErrorCode":0,
        "AuthenticationErrorText":"Success",
        "Position":"10.91.161.165",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"wsapiqa"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted by host."
    },
    "Type":"Refresh"
  }
]
```


5.4.2 Close Automatic Login


```
{
    "Domain":"Login",
    "ID":-1,
    "Type":"Close"
}
```

6 Batch

6.1 Definition of Batch

Client applications can use a single Batch Request to request multiple items. Responses are delivered individually.

The LSEG Real-Time Advanced Distribution Server supports this feature for both snapshot and streaming requests.

When the LSEG Real-Time Advanced Distribution Server receives a Batch Request, it will respond on the same ID of the Request with a Status message that acknowledges receipt of the Batch by indicating a State object with a "Data":"Ok" and "State":"Closed".

The Batch stream closes because all additional responses are provided on individual streams. The ID values of the resulting streams are assigned sequentially according to the order of the entries in the Batch Request's "Key":{"Name":[...]} array, beginning with the ID of the original Batch Request message + 1. The Batch Response and Item Response tiles in the Context section below highlight this exchange.

6.2 Structure

ATTRIBUTE	TYPE	DEFINITION
ID	int,array(int)	Integer value(s) representing the event stream. It can also be used to match the request and responses.
Key	int,array(int)	The key representing the data content or capability requested.
└ Name	string,array(string)	Name(s) of the information requested.

Table 5: Batch Structure

6.3 Context

6.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"256",
            "Position":"127.0.0.1"
        },
        "Name":"user"
    }
}
```

6.3.2 Login Response

```
[
  {
    "Domain":"Login",
    "Elements":{
      "MaxMsgSize":61440,
      "PingTimeout":30
    },
    "ID":1,
    "Key":{
      "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"256",
        "ApplicationName":"ADS",
        "Position":"127.0.0.1",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"user"
},
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted by host."
    },
    "Type":"Refresh"
  }
]
```

6.3.3 Item Request

```
{
    "ID":2,
    "Key":{
        "Name":[
            "TRI.N",
            "IBM.N",
            "T.N"
        ]
    }
}
```

6.3.4 Batch Response


```
[
    {
        "ID":2,
        "State":{
            "Data":"Ok",
            "Stream":"Closed",
            "Text":"Processed 3 total items from Batch Request.    3 Ok."
        },
        "Type":"Status"
    }
]
```


6.3.5 Item Response: TRI.N


```
[
    {
        "Fields":{
            "ACVOL_1":8719016,
            "ADJUST_CLS":392.8,
            "ASK":9000,
            "ASKSIZE":1,
            "ASKXID":"BOS",
            "ASK_MMID1":"BOS",
            "BID":0.01,
            "BIDSIZE":1,
            "BIDXID":"BOS",
            "BID_MMID1":"BOS",
            "BID_NET_CH":null,
            "BID_TICK_1":"⇩",
            "BLKCOUNT":5,
            "BLKVOLUM":116195,
            "CLOSE_ASK":398.1,
            "CLOSE_BID":398.01,
            "CTS_QUAL":"   ",
            "CUM_EX_MKR":"   ",
            "CURRENCY":"USD",
            "EXCHTIM":"21:00:01",
            "EXDIVDATE":null,
            "GV1_FLAG":null,
            "GV1_TEXT":"-",
            "HIGH_1":398.85,
            "HSTCLBDDAT":null,
            "HSTCLSDATE":"2017-11-28",
            "HST_CLOSE":392.8,
            "HST_CLSBID":null,
            "INSCOND":"   ",
            "INSPRC":null,
```

```
            "INSVOL":null,
            "IRGCOND":"132",
            "IRGPRC":398.85,
            "IRGVOL":144,
            "IRGXID":"CIN",
            "LOW_1":394.11,
            "NETCHNG_1":5.35,
            "NUM_MOVES":16844,
            "OFFCL_CODE":null,
            "OFF_CD_IND":"CUS",
            "OPENEXID":"PSE",
            "OPEN_PRC":396.5,
            "OPN_NETCH":3.7,
            "PCTCHNG":1.36,
            "PRCTCK_1":"⇧",
            "PRC_QL2":"   ",
            "PRC_QL_CD":"   ",
            "PREF_DISP":2254,
            "PRNTBCK":968902,
            "PROD_PERM":6560,
            "PROV_SYMB":"GOOG",
            "QUOTIM":"19:01:55",
            "QUOTIM_MS":84932000,
            "RDNDISPLAY":66,
            "RDN_EXCHD2":"NMQ",
            "RDN_EXCHID":"   ",
            "RECORDTYPE":113,
            "SALTIM":"19:01:52",
            "SALTIM_MS":84514000,
            "SEQNUM":1266750,
            "TIMCOR":null,
            "TIMCOR_MS":137197144,
            "TRADE_DATE":"2017-11-29",
            "TRDPRC_1":398.15,
            "TRDTIM_MS":75601000,
            "TRDVOL_1":26506,
            "TRDXID_1":"NAS",
            "TRD_UNITS":"2DP ",
            "TURNOVER":392.8,
            "VOL_X_PRC1":397.9481
        },
        "ID":3,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "QOS":{
            "Rate":"TickByTick",
            "Timeliness":"Realtime"
        },
```

```
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"All is well"
        },
        "Type":"Refresh"
    }
]
```


6.3.6 Item Response: IBM.N


```
[
    {
        "Fields":{
            "ACVOL_1":8719016,
            "ADJUST_CLS":392.8,
            "ASK":9000,
            "ASKSIZE":1,
            "ASKXID":"BOS",
            "ASK_MMID1":"BOS",
            "BID":0.01,
            "BIDSIZE":1,
            "BIDXID":"BOS",
            "BID_MMID1":"BOS",
            "BID_NET_CH":null,
            "BID_TICK_1":"⇩",
            "BLKCOUNT":5,
            "BLKVOLUM":116195,
            "CLOSE_ASK":398.1,
            "CLOSE_BID":398.01,
            "CTS_QUAL":"   ",
            "CUM_EX_MKR":"   ",
            "CURRENCY":"USD",
            "EXCHTIM":"21:00:01",
            "EXDIVDATE":null,
            "GV1_FLAG":null,
            "GV1_TEXT":"-",
            "HIGH_1":398.85,
            "HSTCLBDDAT":null,
            "HSTCLSDATE":"2017-11-28",
            "HST_CLOSE":392.8,
            "HST_CLSBID":null,
            "INSCOND":"   ",
            "INSPRC":null,
            "INSVOL":null,
            "IRGCOND":"132",
            "IRGPRC":398.85,
            "IRGVOL":144,
```

```
            "IRGXID":"CIN",
            "LOW_1":394.11,
            "NETCHNG_1":5.35,
            "NUM_MOVES":16844,
            "OFFCL_CODE":null,
            "OFF_CD_IND":"CUS",
            "OPENEXID":"PSE",
            "OPEN_PRC":396.5,
            "OPN_NETCH":3.7,
            "PCTCHNG":1.36,
            "PRCTCK_1":"⇧",
            "PRC_QL2":"   ",
            "PRC_QL_CD":"   ",
            "PREF_DISP":2254,
            "PRNTBCK":968902,
            "PROD_PERM":6560,
            "PROV_SYMB":"GOOG",
            "QUOTIM":"19:01:55",
            "QUOTIM_MS":84932000,
            "RDNDISPLAY":66,
            "RDN_EXCHD2":"NMQ",
            "RDN_EXCHID":"   ",
            "RECORDTYPE":113,
            "SALTIM":"19:01:52",
            "SALTIM_MS":84514000,
            "SEQNUM":1266750,
            "TIMCOR":null,
            "TIMCOR_MS":137197144,
            "TRADE_DATE":"2017-11-29",
            "TRDPRC_1":398.15,
            "TRDTIM_MS":75601000,
            "TRDVOL_1":26506,
            "TRDXID_1":"NAS",
            "TRD_UNITS":"2DP ",
            "TURNOVER":392.8,
            "VOL_X_PRC1":397.9481
        },
        "ID":4,
        "Key":{
            "Name":"IBM.N",
            "Service":"DF_RMDS"
        },
        "QOS":{
            "Rate":"TickByTick",
            "Timeliness":"Realtime"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"All is well"
```

```
        },
        "Type":"Refresh"
    }
]
```

6.3.7 Item Response: T.N

```
[
    {
        "Fields":{
            "ACVOL_1":8719016,
            "ADJUST_CLS":392.8,
            "ASK":9000,
            "ASKSIZE":1,
            "ASKXID":"BOS",
            "ASK_MMID1":"BOS",
            "BID":0.01,
            "BIDSIZE":1,
            "BIDXID":"BOS",
            "BID_MMID1":"BOS",
            "BID_NET_CH":null,
            "BID_TICK_1":"⇧",
            "BLKCOUNT":5,
            "BLKVOLUM":116195,
            "CLOSE_ASK":398.1,
            "CLOSE_BID":398.01,
            "CTS_QUAL":"   ",
            "CUM_EX_MKR":"   ",
            "CURRENCY":"USD",
            "EXCHTIM":"21:00:01",
            "EXDIVDATE":null,
            "GV1_FLAG":null,
            "GV1_TEXT":"-",
            "HIGH_1":398.85,
            "HSTCLBDDAT":null,
            "HSTCLSDATE":"2017-11-28",
            "HST_CLOSE":392.8,
            "HST_CLSBID":null,
            "INSCOND":"   ",
            "INSPRC":null,
            "INSVOL":null,
            "IRGCOND":"132",
            "IRGPRC":398.85,
            "IRGVOL":144,
            "IRGXID":"CIN",
            "LOW_1":394.11,
            "NETCHNG_1":5.35,
```

```
            "NUM_MOVES":16844,
            "OFFCL_CODE":null,
            "OFF_CD_IND":"CUS",
            "OPENEXID":"PSE",
            "OPEN_PRC":396.5,
            "OPN_NETCH":3.7,
            "PCTCHNG":1.36,
            "PRCTCK_1":"⇧",
            "PRC_QL2":"   ",
            "PRC_QL_CD":"   ",
            "PREF_DISP":2254,
            "PRNTBCK":968902,
            "PROD_PERM":6560,
            "PROV_SYMB":"GOOG",
            "QUOTIM":"19:01:55",
            "QUOTIM_MS":84932000,
            "RDNDISPLAY":66,
            "RDN_EXCHD2":"NMQ",
            "RDN_EXCHID":"   ",
            "RECORDTYPE":113,
            "SALTIM":"19:01:52",
            "SALTIM_MS":84514000,
            "SEQNUM":1266750,
            "TIMCOR":null,
            "TIMCOR_MS":137197144,
            "TRADE_DATE":"2017-11-29",
            "TRDPRC_1":398.15,
            "TRDTIM_MS":75601000,
            "TRDVOL_1":26506,
            "TRDXID_1":"NAS",
            "TRD_UNITS":"2DP ",
            "TURNOVER":392.8,
            "VOL_X_PRC1":397.9481
        },
        "ID":5,
        "Key":{
            "Name":"T.N",
            "Service":"DF_RMDS"
        },
        "QOS":{
            "Rate":"TickByTick",
            "Timeliness":"Realtime"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"All is well"
        },
        "Type":"Refresh"
    }
```

```
]
```

6.3.8 Item Update(s)

```
[
    {
        "Fields":{
            "ASK":401.54,
            "ASKSIZE":10,
            "ASKXID":"NAS",
            "ASK_MMID1":"NAS",
            "BID":401.5,
            "BIDSIZE":18,
            "BIDXID":"NAS",
            "BID_MMID1":"NAS",
            "BID_NET_CH":3.49,
            "BID_TICK_1":"⇩",
            "GV1_TEXT":"-",
            "QUOTIM":"14:40:32:000:000:000",
            "QUOTIM_MS":52832000
        },
        "ID":4,
        "Key":{
            "Name":"IBM.N",
            "Service":"DF_RMDS"
        },
        "Type":"Update",
        "UpdateType":"Quote"
    }
]
```

6.3.9 Batch Close

```
{
    "ID":[
        3,
        4,
        5
    ],
    "Type":"Close"
}
```

6.3.10 Batch Close Response

```
[
    {
        "ID":3,
        "State":{
            "Data":"Ok",
            "Stream":"Closed",
            "Text":"Processed 3 total stream ids from Batch Close Request. 3 Ok. "
        },
        "Type":"Status"
    }
]
```

6.3.11 Close Login

```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

7 Posting

7.1 Definition of Posting

LSEG Real-Time Advanced Distribution Server provides the capability of Posting for Consumer applications to push content into a Cache located in the LSEG Real-Time Distribution System.

7.2 Structure

ATTRIBUTE	TYPE	DEFINITION
Ack	boolean	The provider should acknowledge the message when received and applied.
Domain	string,int	The domain model represented by this message. Defaults to Market Price if absent. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ID	int	Integer value representing the event stream. It can also be used to match the request and responses.
Message	object	A message such as Refresh or Update containing the content that is being posted. See the other types of Message for details.

Table 6: Posting Structure

ATTRIBUTE	TYPE	DEFINITION
└ Domain	string,int	The domain model represented by this message. Defaults to Market Price if absent. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
└ Fields	object	A field list.
└ ID	int,array(int)	Integer value(s), or array of integers representing the event stream. It can also be used to match the request and responses.
└ Type	string,int	The message classification. Defaults to Request if absent. • Ack • Close • Generic • Post • Refresh • Request • Status • Update
PostID	int	Used by upstream devices to distinguish different Post messages. Each Post message in a multi-part post must use the same PostID value.
PostUserInfo	object	Represents information about the posting user.

Table 6: Posting Structure

ATTRIBUTE	TYPE	DEFINITION
└ Address	int	Dotted-decimal string representing the IP Address of the posting user.
└ UserID	int	ID of posting user.
Type	string,int	The message classification. Set to Post for Post message. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 6: Posting Structure

7.3 Context

7.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"256",
            "Position":"127.0.0.1"
        },
        "Name":"user"
    }
}
```

7.3.2 Login Response 1


```
[
    {
        "Domain":"Login",
        "Elements":{
            "MaxMsgSize":61440,
            "PingTimeout":30
        },
        "ID":1,
        "Key":{
            "Elements":{
                "AllowSuspectData":1,
                "ApplicationId":"256",
                "ApplicationName":"ADS",
                "Position":"127.0.0.1",
                "ProvidePermissionExpressions":1,
                "ProvidePermissionProfile":0,
                "SingleOpen":1,
                "SupportBatchRequests":7,
                "SupportEnhancedSymbolList":1,
                "SupportOMMPost":1,
                "SupportOptimizedPauseResume":1,
                "SupportPauseResume":1,
                "SupportStandby":0,
                "SupportViewRequests":1
            },
            "Name":"user"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"Login accepted by host."
        },
        "Type":"Refresh"
    }
]
```


7.3.3 Item Request 1


```
{
  "ID":2,
  "Key":{
    "Name":"TRI.N"
  }
}
```

7.3.4 Item Response


```
[
    {
        "Fields":{
            "ACVOL_1":8719016,
            "ADJUST_CLS":392.8,
            "ASK":9000.0,
            "ASKSIZE":1,
            "ASKXID":"BOS",
            "ASK_MMID1":"BOS",
            "BID":0.01,
            "BIDSIZE":1,
            "BIDXID":"BOS",
            "BID_MMID1":"BOS",
            "BID_NET_CH":null,
            "BID_TICK_1":"⇧",
            "BLKCOUNT":5,
            "BLKVOLUM":116195,
            "CLOSE_ASK":398.1,
            "CLOSE_BID":398.01,
            "CTS_QUAL":"   ",
            "CUM_EX_MKR":"   ",
            "CURRENCY":"USD",
            "EXCHTIM":"21:00:01",
            "EXDIVDATE":null,
            "GV1_FLAG":null,
            "GV1_TEXT":"-",
            "HIGH_1":398.85,
            "HSTCLBDDAT":null,
            "HSTCLSDATE":"2017-11-28",
            "HST_CLOSE":392.8,
            "HST_CLSBID":null,
            "INSCOND":"   ",
            "INSPRC":null,
            "INSVOL":null,
            "IRGCOND":"132",
            "IRGPRC":398.85,
            "IRGVOL":144,
            "IRGXID":"CIN",
            "LOW_1":394.11,
            "NETCHNG_1":5.35,
            "NUM_MOVES":16844,
            "OFFCL_CODE":null,
            "OFF_CD_IND":"CUS",
            "OPENEXID":"PSE",
            "OPEN_PRC":396.5,
            "OPN_NETCH":3.7,
            "PCTCHNG":1.36,
```

```
            "PRCTCK_1":"⇧",
            "PRC_QL2":"   ",
            "PRC_QL_CD":"   ",
            "PREF_DISP":2254,
            "PRNTBCK":968902,
            "PROD_PERM":6560,
            "PROV_SYMB":"GOOG",
            "QUOTIM":"19:01:55",
            "QUOTIM_MS":84932000,
            "RDNDISPLAY":66,
            "RDN_EXCHD2":"NMQ",
            "RDN_EXCHID":"   ",
            "RECORDTYPE":113,
            "SALTIM":"19:01:52",
            "SALTIM_MS":84514000,
            "SEQNUM":1266750,
            "TIMCOR":null,
            "TIMCOR_MS":137197144,
            "TRADE_DATE":"2017-11-29",
            "TRDPRC_1":398.15,
            "TRDTIM_MS":75601000,
            "TRDVOL_1":26506,
            "TRDXID_1":"NAS",
            "TRD_UNITS":"2DP ",
            "TURNOVER":392.8,
            "VOL_X_PRC1":397.9481
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "QOS":{
            "Rate":"TickByTick",
            "Timeliness":"Realtime"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"All is well"
        },
        "Type":"Refresh"
    }
]
```

7.3.5 Item Request 2

```
{
    "ID":2,
    "Key":{
        "Name":"TRI.N"
    }
}
```

7.3.6 Login Response 2

```
[
  {
    "Domain":"Login",
    "Elements":{
      "MaxMsgSize":61440,
      "PingTimeout":30
    },
    "ID":1,
    "Key":{
      "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"256",
        "ApplicationName":"ADS",
        "Position":"127.0.0.1",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"user"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted by host."
    },
    "Type":"Refresh"
  }
]
```

7.3.7 Item Request 3

```
{
    "ID":2,
    "Key":{
        "Name":"TRI.N"
    }
}
```

7.3.8 Item Response 2


```
[
    {
        "Fields":{
            "ACVOL_1":8719016,
            "ADJUST_CLS":392.8,
            "ASK":9000,
            "ASKSIZE":1,
            "ASKXID":"BOS",
            "ASK_MMID1":"BOS",
            "BID":0.01,
            "BIDSIZE":1,
            "BIDXID":"BOS",
            "BID_MMID1":"BOS",
            "BID_NET_CH":null,
            "BID_TICK_1":"⇧",
            "BLKCOUNT":5,
            "BLKVOLUM":116195,
            "CLOSE_ASK":398.1,
            "CLOSE_BID":398.01,
            "CTS_QUAL":"   ",
            "CUM_EX_MKR":"   ",
            "CURRENCY":"USD",
            "EXCHTIM":"21:00:01",
            "EXDIVDATE":null,
            "GV1_FLAG":null,
            "GV1_TEXT":"-",
            "HIGH_1":398.85,
            "HSTCLBDDAT":null,
            "HSTCLSDATE":"2017-11-28",
            "HST_CLOSE":392.8,
            "HST_CLSBID":null,
            "INSCOND":"   ",
            "INSPRC":null,
            "INSVOL":null,
            "IRGCOND":"132",
            "IRGPRC":398.85,
            "IRGVOL":144,
            "IRGXID":"CIN",
            "LOW_1":394.11,
            "NETCHNG_1":5.35,
            "NUM_MOVES":16844,
            "OFFCL_CODE":null,
            "OFF_CD_IND":"CUS",
```

```
        },
            "Type":"Refresh"
    }
]
```

7.3.9 Item Update

```
[
    {
        "Fields":{
            "ASK":401.54,
            "ASKSIZE":10,
            "ASKXID":"NAS",
            "ASK_MMID1":"NAS",
            "BID":401.5,
            "BIDSIZE":18,
            "BIDXID":"NAS",
            "BID_MMID1":"NAS",
            "BID_NET_CH":3.49,
            "BID_TICK_1":"\u00fe",
            "GV1_TEXT":"-",
            "QUOTIM":"14:40:32:000:000:000",
            "QUOTIM_MS":52832000
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "Type":"Update",
        "UpdateType":"Quote"
    }
]
```

7.3.10 Post

```
{
    "Ack":true,
    "Domain":"MarketPrice",
    "ID":2,
    "Message":{
        "Domain":"MarketPrice",
        "Fields":{
            "ASK":45.57,
            "ASKSIZE":19,
            "BID":45.55,
            "BIDSIZE":18
        },
        "ID":0,
        "Type":"Update"
```

```
    },
    "PostID":1,
    "PostUserInfo":{
        "Address":"127.0.0.1",
        "UserID":10000
    },
    "Type":"Post"
}
```

7.3.11 Ack

```
[
  {
    "AckID":1,
    "ID":2,
    "Text":"All is well",
    "Type":"Ack"
  }
]
```

7.3.12 Close Item

```
{
  "ID":2,
  "Type":"Close"
}
```

7.3.13 Close Login

```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

8 View

8.1 Definition of View

The Views feature allows the client application to request specific fields from a (Level 1) record. Views can be also be applied to Batch requests.

The LSEG Real-Time Advanced Distribution Server supports this feature for both snapshot and streaming requests.

8.2 Structure

ATTRIBUTE	TYPE	DEFINITION
ID	int,array(int)	Integer value(s) representing the event stream. It can also be used to match the request and responses.
Key	int,array(int)	The key representing the data content or capability requested.
└ Name	string,array(string)	Name(s) of the information requested.
View	array(string,number)	An array of field names or IDs that the client application would like to specifically request.

Table 7: View Structure

8.3 Context

8.3.1 Login

```
{
    "Domain":"Login",
    "ID":1,
    "Key":{
        "Elements":{
            "ApplicationId":"256",
            "Position":"127.0.0.1"
        },
        "Name":"user"
    }
}
```

8.3.2 Login Response


```
[
  {
    "Domain":"Login",
    "Elements":{
      "MaxMsgSize":61440,
      "PingTimeout":30
    },
    "ID":1,
    "Key":{
      "Elements":{
        "AllowSuspectData":1,
        "ApplicationId":"256",
        "ApplicationName":"ADS",
        "Position":"127.0.0.1",
        "ProvidePermissionExpressions":1,
        "ProvidePermissionProfile":0,
        "SingleOpen":1,
        "SupportBatchRequests":7,
        "SupportEnhancedSymbolList":1,
        "SupportOMMPost":1,
        "SupportOptimizedPauseResume":1,
        "SupportPauseResume":1,
        "SupportStandby":0,
        "SupportViewRequests":1
      },
      "Name":"user"
    },
    "State":{
      "Data":"Ok",
      "Stream":"Open",
      "Text":"Login accepted by host."
    },
    "Type":"Refresh"
  }
]
```


8.3.3 View Item Request


```
{
    "ID":2,
    "Key":{
        "Name":"TRI.N"
    },
    "View":[
        "BID",
        "ASK",
        "BIDSIZE"
    ]
}
```

8.3.4 Item Response

```
[
    {
        "Fields":{
            "ASK":9000,
            "BID":0.01,
            "BIDSIZE":1
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "QOS":{
            "Rate":"TickByTick",
            "Timeliness":"Realtime"
        },
        "State":{
            "Data":"Ok",
            "Stream":"Open",
            "Text":"All is well"
        },
        "Type":"Refresh"
    }
]
```

8.3.5 Item Update(s)

```
[
    {
        "Fields":{
            "ASK":401.54,
            "BID":401.5,
            "BIDSIZE":18
        },
        "ID":2,
        "Key":{
            "Name":"TRI.N",
            "Service":"DF_RMDS"
        },
        "Type":"Update",
        "UpdateType":"Quote"
    }
]
```

8.3.6 Close Item

```
{
    "ID":2,
    "Type":"Close"
}
```

8.3.7 Close Login

```
{
    "Domain":"Login",
    "ID":1,
    "Type":"Close"
}
```

9 Examples

9.1 Language-Specific Examples

LSEG provides examples written in C#, Go, Java, Node.js, Perl, Python, and R that illustrate how retrieve data. There are several examples written in each language: Market Price, Batch View Request for Market Price, Posting, and Ping. These examples and README files with their descriptions are available on GitHub at the URL https://github.com/Refinitiv/websocket-api/tree/master/Applications/Examples.

9.2 Delivery Platform Connectivity Examples

On GitHub, LSEG provides a set of examples showcasing Delivery Platform connectivity. These examples (provided in Java, Python, and CSharp) show how to perform Service Discovery and Delivery Platform Authentication. For further details on Delivery Platform connectivity, refer to the *Real-Time - Optimized Installation and Configuration Guide* accessible at the URL:

https://developers.lseg.com/content/dam/devportal/realtimeapi_pdfs/rrt_optimized.pdf.

You can access Delivery Platform examples at the URL: https://github.com/Refinitiv/websocket-api/tree/master/Applications/Examples/LDP.

10 Primitive Types

While JSON data types are primarily used to represent data in the WebSocket API protocol, in some messages certain Primitive types may also be defined as the value of a "Type" name to describe message data.

A Primitive type represents some type of base, system information (such as integers, dates, or strings).

TYPE	DEFINITION
Array	An array containing values whose Type is one of the other primitives in this section. All values in the array have the same Type.
AsciiString	An ASCII string which should contain only characters that are valid in ASCII specification. Represented in JSON by a string.
Buffer	Represents a raw byte buffer type. Represented in JSON by a string, which uses a Base64 encoding.
Date	Defines a date with month, day, and year values. Represented in JSON by a string. Date follows the ISO 8601 format for representing date.
DateTime	Combined representation of date and time. Contains all members of Date and Time. Represented in JSON by a string. DateTime follows the ISO 8601 format for representing the date and time.
Double	A double-precision floating-point type. Represented in JSON by a number or a string with one of the following values: • "Inf": Infinity • "-Inf": Negative Infinity • "NaN": Not a Number
Enum	A string or integer, depending on whether a given Enum value has a string to represent it.
Float	A single-precision floating-point type. Represented in JSON by a number or a string with one of the following values: • "Inf": Infinity • "-Inf": Negative Infinity • "NaN": Not a Number
Int	A signed integer type. Represented in JSON by a number.
Real	A representation of a decimal or fractional value. Represented in JSON by a number, or a string with one of the following values: • "Inf": Infinity • "-Inf": Negative Infinity • "NaN": Not a Number
RmtesString	Represents an RMTES (a multilingual text encoding standard) string. An RMTES string is represented in the WebSocket API Protocol as a UTF-8 encoded string.
	NOTE: The LSEG Real-Time Advanced Distribution Server converts RMTES strings into UTF-8 strings. Strings posted into the LSEG Real-Time Distribution System with an RMTES type should follow the ASCII character set.
Time	Defines a time with hour, minute, second, millisecond, microsecond, and nanosecond values. Represented in JSON by a string. Date follows the ISO 8601 format for representing time.

Table 8: Primitive Types

TYPE	DEFINITION
UInt	An unsigned integer type. Represented in JSON by a number.
Utf8String	Represents a UTF8 string which should follow the UTF8 encoding standard and contain only characters valid within that set. Represented in JSON by a string.

Table 8: Primitive Types

11 Container: Elements

The Elements container is represented by a JSON object containing a series of elements. An Element's value may be a JSON string, number or object.

- If the Element value is a JSON number then the Open Message Model primitive type is assumed to be of type UInt.
- If the Element value is a JSON string then the Open Message Model primitive type is assumed to be of type AsciiString.
- If the Element is a JSON object it will be made up of Type and Data attributes, where the Type attribute specifies the Open Message Model Primitive or Container type and the Data attribute contains the data of the element.
- If the content of an AsciiString or UInt Element is null or empty, it must be encoded as a JSON object so that the type can be determined.

```
"Elements": {
    "AllowSuspectData": 1,
    "ApplicationName": "ADS",
    "ApplicationId": {
        "Type": "UInt",
        "Data": null
    }
}
```

```
"Elements": {
    "AllowSuspectData": null,
    "ApplicationName": null,
    "ApplicationId": null
}
```

A blank, or empty, Element is represented by the JSON keyword null.

```
"Elements": null
```

12 Container: Fields

The Fields container is represented by a JSON object containing a series of field value pairs.

The name attribute is the name of the field as defined by the field dictionary in use by the LSEG Real-Time Advanced Distribution Server. The value of the attribute may be a numeric, string, or other container type.

For example:

```
"Fields": {
    "BID": 40.74,
    "ASK": 40.75,
    "BIDSIZE": 63,
    "ASKSIZE": 223,
    "QUOTIM": "19:50:05:000:000:000"
}
```

Another example with fields set to `null`:

```
"Fields": {
    "BID": null,
    "ASK": null,
    "BIDSIZE": null,
    "ASKSIZE": null,
    "QUOTIM": null
}
```

A blank, or empty, **Fields** attribute is represented by the JSON keyword `null`.

```
"Fields": null
```

13 Container: Json

The Json container is represented by any standard format JSON object.

```
"Json": {
    "Hello World": "This is my JSON data"
}
```

14 Container: Map

The Map container is represented by a JSON object containing a series of associated key-value pairs.

A Map entry can contain a Fields or Elements container, and all map entries must contain the same type of container, unless the entry's action is Delete (in which case it will not contain content).

All keys for a map are the type given by the Map's KeyType attribute.

14.1 Members

ATTRIBUTE	DEFINITION
CountHint	Optional. An approximate count of MapEntries that will be present in the map. This is typically used when splitting entries across a multi-part response (available with Refresh, Generic, or Post messages). (Default: **0**).
Entries	Optional. Contains the Entries of the Map. (Default: **0**).
└ Action	**Required**. Action to use when applying the information in this entry. • "Add" • "Delete" • "Update"
└ Key	**Required**. Key value associated with this entry.
└ PermData	Optional. Includes any permission data associated with the entry (Default: **0**)
KeyFieldID	Optional. If present, indicates that the Key of each entry is the content of a field, and which field that content represents. (Default: **0**).
KeyType	**Required**. Indicates the type of key that appears on each entry in this Map. See the Primitives section for details.
Summary	Conveys information that applies to every entry housed in the container. This eliminates unnecessary data repetition by sending it once, instead of including such data in each entry.

Table 9: Map Members

14.2 For Example

```
"Map":{
    "Entries":[
        {
            "Action":"Add",
            "Fields":{
                "ORDER_PRC": 326.3,
                "ORDER_SIDE":1,
                "ORDER_SIZE": 100,
                "QUOTIM_MS": 78398067
            },
            "Key":"ABCDEFGHIJKL"
        }
    ],
    "KeyType":"Buffer",
    "KeyFieldID": 3426
}
```

15 Container: Message

The Message container is represented by a JSON object containing a series of one or more key-value pairs representing Request, Refresh, Update, Status, Close, Post, Ack, and Generic Messages. See the definition of each message type for a list of valid keys permitted for that type. A Message is encoded using this outermost JSON construct: {message}. If the “Type” key is unspecified in a Message, it is considered a Request Message.

If that outermost construct is a JSON array, it is considered a packed message: [{message 1}, {message 2}, ...]. A packed message may consist of any of the various message types defined in this document and may contain all, or, at a minimum, the required fields of each message type.

To send a message in a message (see Post Message below), the “Message” key, where the value is a message, may be used.

15.1 Request Message

```
{
    "ID": 2,
    "Key": {
        "Name": "TRI.N"
    }
}
```

15.2 Packed Request Messages

```
[
    {
        "ID":3,
        "Key":{
            "Name":"IBM.N"
        }
    },
    {
        "ID":2,
        "Key":{
            "Name":"TRI.N"
        }
    }
]
```

15.3 Post Message

```
{
    "Ack":true,
    "Domain":"MarketPrice",
    "ID":2,
    "Message":{
        "Domain":"MarketPrice",
        "Fields":{
            "ASK":45.57,
            "ASKSIZE":19,
            "BID":45.55,
            "BIDSIZE":18
        },
        "ID":0,
        "Type":"Update"
    },
    "PostID":1,
    "PostUserInfo":{
        "Address":"127.0.0.1",
        "UserID":55555
    },
    "Type":"Post"
}
```

16 Container: Opaque

The Opaque container is supported as a base container or part of other complex container types (e.g. Elements, Fields). The JSON data of a Opaque container distributed from the LSEG Real-Time Distribution System is always base64 encoded.

```
"Opaque": "V2ViU29ja2V0IEFQSSB3YXMgaGVyZQ=="
```

A blank, or empty, Opaque attribute is represented by the JSON keyword null.

```
"Opaque": null
```

17 Container: Series

The Series container is represented by a JSON object. Summary data is expected to be in the same data format as the Series Entries, and all Series Entries must be the same data format. A Series entry can contain a Fields or Elements container. Summary data is expected to be the same data format. CountHint and Summary are optional.

17.1 Members

ATTRIBUTE	DEFINITION
CountHint	An approximate count of Series Entries that will be present in the Series. This is typically used when splitting entries across a multi-part response (available with Refresh, Generic, or Post messages). CountHint defaults to **0**.
Entries	Contains the Entries of the Series. Entries Defaults to **0**.
Summary	Conveys information that applies to every entry housed in the container. This eliminates unnecessary data repetition by sending it once, instead of including such data in each entry.

Table 10: Map Members

17.2 For Example

```
"Series" : {
    "Summary": {
        "Fields":{
            "BID":45.01,
            "BIDSIZE":18.77
        }
    },
    "CountHint":10,
    "Entries": [
        {
            "BID":45.55,
            "BIDSIZE":18,
            "ASK":45.57,
            "ASKSIZE":19
        },
        {
            "BID":55.55,
            "BIDSIZE":28,
            "ASK":55.57,
            "ASKSIZE":29
        }
    ]
}
```

18 Container: Vector

The Vector container is represented by a JSON object containing a series of index-value paired entries. Summary data should be in the same data format as the Entries. A Vector entry can contain a Fields or Elements container. Summary, CountHint, SupportSorting, and PermData per entry are optional.

18.1 Members

ATTRIBUTE	DEFINITION
CountHint	An approximate count of Vector Entries that will be present in the Vector. This is typically used when splitting entries across a multi-part response (available with Refresh, Generic, or Post messages). CountHint defaults to 0.
Entries	Contains the entries of the Vector. Entries Defaults to 0.
└ Action	**Required**. Specifies the action to use when applying the information in this entry. • "Set" • "Update" • "Clear" • "Insert" • "Delete"
└ Index	**Required**. Index value associated with this entry.
└ PermData	Permission Data associated with the entry. PermData defaults to 0.
Summary	Conveys information that applies to every entry housed in the container. This eliminates unnecessary data repetition by sending it once, instead of including such data in each entry.
SupportSorting	Indicates whether to support sorting in the Vector. SupportSorting defaults to false.

Table 11: Vector Members

18.2 For Example

```
"Vector" : {
    "Summary": {
        "Fields":{
            "BID":45.01,
            "BIDSIZE":18.77
        }
    },
    "CountHint":2,
    "SupportSorting":True,
    "Entries": [
        {
            "Index":1,
            "Action":"Update",
            "PermData":"acac",
            "Fields": {
```

```
                "BID":45.55,
                "BIDSIZE":18,
                "ASK":45.57,
                "ASKSIZE":19
            }
        },
        {
            "Index":2,
            "Action":"Update",
            "Fields": {
                "BID":55.55,
                "BIDSIZE":28,
                "ASK":55.57,
                "ASKSIZE":29
            }
        }
    ]
}
```

19 Container: Xml

The Xml container is supported as a base container or part of other complex container types (e.g. Elements, Fields). In order to comply with standard JSON format, characters such as " must be escaped with \ characters.

```
"Xml": "XML data in JSON"
```

```
"Xml": "The computer said, \"Hello World!\""
```

A blank, or empty, Fields attribute is represented by the JSON keyword null.

```
"Xml": null
```

20 Messages: Ack Message

20.1 Ack Message Description

The Ack message is used to acknowledge an outstanding request or close.

20.2 Ack Message Structure

ATTRIBUTE	TYPE	DEFINITION
AckID	int	Used to associate this Ack with the message it is acknowledging.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ExtHdr	string	An optional extension to the request message in case an attribute is identified that currently doesn't fit into the request message header.
ID	int	**Required.** Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or requested capability.
└ Elements	object	An Element List describing additional attributes of the item stream.

Table 12: Ack Message Structure

ATTRIBUTE	TYPE	DEFINITION
Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
Name	string,array(string)	Name(s) of the information requested.
NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
NakCode	string,int	Specifies the NAK code. • AccessDenied: The user is not permissioned to post on the item or service. • DeniedBySrc: The source being posted to has denied accepting this post message. • GatewayDown: A gateway device for handling posted or contributed information is down or unavailable. • InvalidContent: The content of the post message is invalid (it does not match the expected formatting) and cannot be posted. • None: No Nak Code. • NoResources: Some component along the path of the post message does not have appropriate resources available to continue processing the post. • NoResponse: There is no response from the source being posted to. This may mean that the source is unavailable or that there is a delay in processing the posted information. • NotOpen: The item being posted to does not have an available stream. • SourceDown: The source being posted to is down or unavailable. • SourceUnknown: The source being posted to is unknown and unreachable. • SymbolUnknown: The system does not recognize the item information provided with the post message. This may be an invalid item.
Private	boolean	Specifies whether the stream is stream. If absent, Private defaults to false.
Qualified	boolean	Specifies whether the stream is qualified. If absent, Qualified defaults to false.
SeqNumber	int	Sequence number intended to help with temporal ordering. Typically, this will be incremented with every message, but may have gaps depending on the sequencing algorithm being used.
Text	string	Provides additional information about the acceptance or rejection of the message being acknowledged.

Table 12: Ack Message Structure

ATTRIBUTE	TYPE	DEFINITION
Type	string,int	**Required.** Specifies the message classification. For Ack messages, Type is Ack. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 12: Ack Message Structure

21 Close

21.1 Close Message Description

The Close message is used to cancel an outstanding request or to stop an existing event stream.

21.2 Close Message Structure

ATTRIBUTE	TYPE	DEFINITION
Domain	string,int	The domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ID	int,array(int)	**Required.** Integer value(s) representing the stream(s) to close.

Table 13: Close Message Structure

ATTRIBUTE	TYPE	DEFINITION
Type	string,int	**Required.** The message classification. For Close messages, Type is Close. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 13: Close Message Structure

22 Error Message

22.1 Error Message Description

An Error message is received when a client sends JSON that is invalid with the WebSocket API. The Error message contains information regarding what caused the error.

Examples of common Error messages are listed below.

22.2 Error Message Structure

ATTRIBUTE	TYPE	DEFINITION
Debug	object	An object containing additional information about the Error in order to help with debugging. Optional.
└ File	string	The file where the error occurred.
└ Line	int	The line where the error occurred.
└ Message	string	The JSON message that caused the error.
└ Offset	int	The location of where the error occurred in the JSON message that caused the error.
ID	int	**Required.** Integer value representing the stream of the message that caused the error (or 0 if ID was nonrecoverable).
Text	string	A message that provides details of the error.
Type	string,int	**Required.** Specifies the message classification. For Error messages, Type is Error. • Ack • Close • Error • Generic • Ping • Pong • Post • Refresh • Request • Status • Update

Table 14: Error Message Structure

22.3 Error: Unexpected Token Type

A key in a JSON Object has a value of an unexpected data type.

ID Contains a String

```
SENT:
{
    "ID": "2",
    "Key": {
        "Name": "TRI.N"
    }
}

RECEIVED:
[
    {
        "ID": 0,
        "Type": "Error",
        "Text": "JSON Converter Token Type error: Expected 'PRIMITIVE' for key 'ID' Received
                'STRING'",
        "Debug": {
            "File": "Converter/jsonToRwfSimple.C",
            "Line": 326,
             "Offset": 11,
            "Message": "{\n    \"ID\": \"2\",\n    \"Key\": {\n        \"Name\": \"TRI.N\"\n    }\n}"
        }
    }
]
```

22.4 Error: Unexpected Parameter

A key in a JSON Object has an unexpected value.

"Type" Contains an Unexpected Value of "ExtraInfo"

```
SENT:
{
    "ID": 2,
    "Type": "ExtraInfo",
    "Key": {
        "Name": "TRI.N"
    }
}

RECEIVED:
[
    {
        "ID": 2,
        "Type": "Error",
        "Text": "JSON Unexpected Value. Received 'ExtraInfo' for key 'Type'",
        "Debug": {
            "File": "Converter/jsonToRwfSimple.C",
            "Line": 441,
             "Offset": 23,
            "Message": "{\n    \"ID\": 2,\n\t\"Type\": \"ExtraInfo\",\n    \"Key\": {\n
                    \"Name\": \"TRI.N\"\n    }\n}"
        }
    }
]
```

22.5 Error: Missing Key

A required key is missing for a particular JSON Object.

Required "ID" (for Request Messages) is Missing

```
SENT:
{
    "Key": {
        "Name": "TRI.N"
    }
}

RECEIVED:
[
    {
        "ID": 0,
        "Type": "Error",
        "Text": "JSON Missing required key 'ID'",
        "Debug": {
            "File": "Converter/jsonToRwfSimple.C",
            "Line": 965,
            "Message": "{\n    \"Key\": {\n        \"Name\": \"TRI.N\"\n    }\n}"
        }
    }
]
```

22.6 Error: Unexpected Key

An unexpected key is present for a particular JSON Object.

NOTE: This error is caught ONLY when the following configuration parameter is present: ***ads*catchUnknownJsonKeys: True**.

An Unexpected Key of "PlaceHolder" is Present

```
SENT:
{
    "ID": 2,
    "PlaceHolder": "1",
    "Key": {
        "Name": "TRI.N"
    }
}

RECEIVED:
[
    {
        "ID": 2,
        "Type": "Error",
        "Text": "JSON Unexpected Key. Received 'PlaceHolder'",
        "Debug": {
            "File": "Converter/jsonToRwfSimple.C",
            "Line": 1383,
            "Offset": 16,
            "Message": "{\n    \"ID\": 2,\n    \"PlaceHolder\": 1,\n    \"Key\": {\n        \"Name\":
                    \"TRI.N\"\n    }\n}"
        }
    }
]
```

22.7 Error: Unexpected Field IDentifier

An unexpected key is present for a particular JSON Object.

NOTE: This error is on by default, however it can be disabled when the following parameter is present in the **ads_pop.cnf** file: ***ads*catchUnknownJsonFids: False**.

"Fields" Contains Unexpected Field IDentifier "BID_CUSTOM"

```
SENT:
{
    "Type": "Post",
    "Message": {
        "Type": "Update",
        "Fields": {
            "ASKSIZE": 19,
            "ASK": 0,
            "BIDSIZE": 18,
            "BID_CUSTOM": 45.55
        },
        "ID": 0,
        "Domain": "MarketPrice",
        "Key": {
            "Service": 60000,
            "Name": "TRI.N"
        }
    },
    "Ack": true,
    "PostUserInfo": {
        "Address": 000000000,
        "UserID": 1
    },
    "ID": 3,
    "Domain": "MarketPrice",
    "Key": {
        "Service": 257,
        "Name": "TRI.N"
    },
    "PostID": 2
}
```

```
RECEIVED:
[
    {
        "ID": 3,
        "Type": "Error",
        "Text": "JSON Unexpected FID. Received 'BID_CUSTOM' for key 'Fields'",
        "Debug": {
            "File": "Converter/jsonToRwfSimple.C",
            "Line": 4460,
            "Offset": 138,
            "Message": "{\n   \"Type\": \"Post\",\n   \"Message\": {\n   \"Type\":
                \"Update\",\n   \"Fields\": {\n   \"ASKSIZE\": 19,\n   \"ASK\": 0,\n
                \"BIDSIZE\":18,\n   \"BID_CUSTOM\": 45.55\n   },\n   \"ID\": 0,\n
                \"Domain\": \"MarketPrice\",\n   \"Key\": {\n   \"Service\": 60000,\n
                \"Name\": \"TRI.N\"\n   }\n   },\n   \"Ack\": true,\n   \"PostUserInfo\":
                {\n        \"Address\": 000000000,\n        \"UserID\": 1\n    },\n    \"ID\": 3,\n
                \"Domain\": \"MarketPrice\",\n   \"Key\": {\n   \"Service\": 257,\n
                \"Name\": \"TRI.N\"\n   },\n   \"PostID\": 2\n}"
        }
    }
]
```

22.8 Error: Array Type Mismatch

The JSON representation of an Open Message Model Array contains different data types within the "Data" JSON Array.

"Data" Array Contains Both Ints and a String

```
SENT:
{
    "ID": 2,
    "Key": {
        "Name": "TRI.N"
    }
    "Map": {
        "Entries": [
            {
                "Action": "Add",
                "Fields": {
                    "ORDER_PRC": 326.3,
                    "ORDER_SIDE": 1,
                    "ORDER_SIZE": 100,
                    "QUOTIM_MS": 78398067
                },
                "Key": {
                    "Type": "Int",
                    "Length": 3,
                    "Data": [
                        1,
                        2,
                        "3"
                    ]
                }
            }
        ]
        "KeyType": "Array"
    }
}
```

```
RECEIVED:
[
    {
        "ID": 2,
        "Type": "Error",
        "Text": "JSON Mixed Types in OMM Array: Received 'PRIMITIVE' and 'STRING' for key
                'Data'",
        "Debug": {
            "File": "Converter/jsonToRwfSimple.C",
            "Line": 5809,
            "Offset": 387,
            "Message": "{\n   \"ID\": 2,\n   \"Key\": {\n   \"Name\": \"TRI.N\"\n   },\n
                    \"Map\": {\n   \"Entries\": [\n   {\n   \"Action\": \"Add\",\n
                    \"Fields\": {\n   \"ORDER_PRC\": 326.3,\n   \"ORDER_SIDE\": 1,\n
                    \"ORDER_SIZE\": 100,\n   \"QUOTIM_MS\": 78398067\n   },\n   \"Key\":
                    {\n   \"Type\": \"Int\",\n   \"Length\": 3,\n   \"Data\": [\n   1,
                    \n   2,\n   \"3\"\n   ]\n   }\n   }\n   ],\n   \"KeyType\":
                    \"Array\"\n   }\n}"
        }
    }
]
```

23 Generic Message

23.1 Generic Message Description

A Generic message is a bi-directional Message that does not have any implicit interaction semantics associated with it.

23.2 Generic Message Structure

ATTRIBUTE	TYPE	DEFINITION
Complete	boolean	Indicates that the payload data in the response is complete. Some domain models require a single response with payload data; others allow multi-part responses of payload data that will have this flag set in the last message. If absent, Complete defaults to true.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ExtHdr	string	An optional extension to the request message in case an attribute is identified that currently doesn't fit into the request message header.
ID	int	**Required.** Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or capability of the Generic message.

Table 15: Generic Message Structure

ATTRIBUTE	TYPE	DEFINITION
└ Elements	object	An Element List describing additional attributes of the item stream.
└ Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
└ Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
└ Name	string,array(string)	Name(s) of the information requested.
└ NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
└ Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
PartNumber	int	Used with multi-part messages. The initial part should use the number 0, and each subsequent part should increment the previous PartNumber by 1.
PermData	string	Contains permission authorization information for all content provided on this stream.
SecSeqNumber	int	An additional user-defined sequence number. Often used as an acknowledgment sequence number.
SeqNumber	int	Sequence number intended to help with temporal ordering. Typically, this will be incremented with every message, but may have gaps depending on the sequencing algorithm being used.
Type	string,int	**Required.** Specifies the message classification. For Generic messages, Type is Generic. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 15: Generic Message Structure (Continued)

24 Ping and Pong Messages

24.1 Ping and Pong Message Descriptions

Ping and Pong messages are exchanged between endpoints of a connection to verify that the remote endpoint is still alive.

The Ping message may be sent by either endpoint. When either endpoint receives a Ping message, it should send a Pong message in response.

The LSEG Real-Time Advanced Distribution Server will send Ping messages to applications when it does not receive traffic for a period of time, so applications must be prepared to respond with a Pong message whenever they receive a Ping. Applications may likewise send Ping messages to elicit Pong messages from the LSEG Real-Time Advanced Distribution Server, but are not required to do so.

The LSEG Real-Time Advanced Distribution Server includes an informational PingTimeout element in its Login response, indicating the time (in seconds) after which the LSEG Real-Time Advanced Distribution Server will disconnect the application if it receives no traffic in response to a sent Ping.

24.2 Message Structure

ATTRIBUTE	TYPE	DEFINITION
Type	string,int	**Required.** Defines the message class. • For Ping messages, set Type to Ping. • For Pong messages, set Type to Pong.

Table 16: Login Structure

25 Post Message

25.1 Post Message Description

The Post message is used to push content into a cache located in the LSEG Real-Time Distribution System.

25.2 Post Message Structure

ATTRIBUTE	TYPE	DEFINITION
Ack	boolean	The provider should acknowledge the message when received and applied. If absent, Ack defaults to false.
Complete	boolean	Indicates that the payload data in the post is complete. Some domain models require a single post with payload data; others allow multi-part post of payload data that will have this flag set in the last message. If absent, Complete defaults to true.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ExtHdr	string	An optional extension to the request message in case an attribute is identified that currently doesn't fit into the request message header.
ID	int	**Required.** Integer value representing the event stream. It can also be used to match the request and responses.

Table 17: Post Message Structure

ATTRIBUTE	TYPE	DEFINITION
Key	object	The key representing the data content or posted capability.
└ Elements	object	An Element List describing additional attributes of the item stream.
└ Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
└ Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
└ Name	string,array(string)	Name(s) of the information requested.
└ NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
└ Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
Message	object	A message such as Refresh or Update containing the content that is being posted. See the other types of Message for details.
PartNumber	int	Used with multi-part messages. The initial part should use the number 0, and each subsequent part should increment the previous PartNumber by 1.
PermData	string	Contains permission authorization information for all content provided on this stream.
PostID	int	Used by upstream devices to distinguish different Post messages. Each Post message in a multi-part post must use the same PostID value.
PostUserInfo	object	Represents information about the posting user.
└ Address	string	**Required**. Dotted-decimal string representing the IP Address of the posting user.
└ UserID	int	**Required**. Specifies the ID of the posting user.
PostUserRights	int	Conveys the rights or abilities of the user posting this content.
SeqNumber	int	Sequence number intended to help with temporal ordering. Typically, this will be incremented with every message, but may have gaps depending on the sequencing algorithm being used.

Table 17: Post Message Structure (Continued)

ATTRIBUTE	TYPE	DEFINITION
Type	string,int	**Required.** Specifies the message classification. For Post messages, Type is Post. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 17: Post Message Structure (Continued)

26 Refresh Message

26.1 Refresh Message Description

A Request message is sent from a consumer to a provider when it wants to request some data, or a capability, available from the provider.

It can also be used to obtain a new response (e.g. synchronization point) or change selected attributes (e.g. priority) for an already open event stream.

26.2 Refresh Message Structure

ATTRIBUTE	TYPE	DEFINITION
ClearCache	boolean	An indication that any previous last value payload data cache for the event stream needs to be deleted. If absent, ClearCache defaults to true.
Complete	boolean	Indicates that the payload data in the response is complete. Some domain models require a single response with payload data; others allow multi-part responses of payload data that will have this flag set in the last message. If absent, Complete defaults to true.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ExtHdr	string	An optional extension to the request message in case an attribute is identified that currently doesn't fit into the request message header.

Table 18: Refresh Message Structure

ATTRIBUTE	TYPE	DEFINITION
ID	int	**Required.** Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or requested capability.
└ Elements	object	An Element List describing additional attributes of the item stream.
└ Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
└ Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
└ Name	string,array(string)	Name(s) of the information requested.
└ NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
└ Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
PartNumber	int	Specifies the part number of the message when part of a multi-part refresh. 0 indicates the message is the first part of the multi-part refresh. Each subsequent part increments PartNumber by 1. This attribute's range is 0 to 32767.
PermData	string	Contains permission authorization information for all content provided on this stream.
PostUserInfo	object	Represents information about the posting user.
└ Address	string	**Required**. Dotted-decimal string representing the IP Address of the posting user.
└ UserID	int	**Required**. Specifies the ID of the posting user.
Private	boolean	Private stream. If absent, Private defaults to false.

Table 18: Refresh Message Structure (Continued)

ATTRIBUTE	TYPE	DEFINITION
Qos	object	Specifies the ***Quality of Service***. Provides classification of data/events to provide different tiers of service. • When specified on the *originating request* without a WorstQos member, Qos is the only Quality of Service for the stream (if the Quality of Service is unavailable, the stream is not opened). • When specified with a WorstQos on the *originating request*, Qos is the best in the range of allowable Quality of Services. When a Quality of Service range is specified, any Quality of Service within the range is acceptable for servicing the stream. • Absence of both Qos nor WorstQos in the *originating request* indicates that any available Quality of Service will satisfy the request. • Some components may require Qos on the initial request and reissue messages. For details, refer to specific component documentation.
└ Dynamic	boolean	Specifies whether or not Qos is dynamic. If absent, KeyInUpdates defaults to false.
└ Rate	string,int	**Required**. Maximum period of change in data (for streaming events). • JitConflated: Just-in-time conflated, meaning that quality is typically tick-by-tick, but if burst data occurs (or if a component cannot keep up with tick-by-tick delivery), multiple updates are combined into a single update to reduce traffic. This value is usually considered a lower quality than TickByTick. • TickByTick: Tick-by-Tick, meaning data is sent for every update. This is the highest quality of rate value. The best overall Quality of Service is a Rate of TickByTick and a Timeliness of Realtime. • TimeConflated: The interval of time (usually in milliseconds) over which data are conflated is provided in RateInfo. This is a lower quality than TickByTick and at times even lower than JitConflated.
└ RateInfo	int	Specifies any information related to Rate.
└ TimeInfo	int	Information related to Timeliness.
└ Timeliness	string,int	**Required**. Specifies the data's age. • Delayed: Timeliness is delayed and the amount of delay is provided in TimeInfo. This is lower quality than Realtime and might be better than DelayedUnknown. • DelayedUnknown: Timeliness is delayed, although the amount of delay is unknown. This is a lower quality than Realtime and might be worse than Delayed (in which case the delay is known). • Realtime: Data is updated as soon as new data is available, and is the highest-quality Timeliness value. In conjunction with a Rate of TickByTick, Realtime is the best overall Quality of Service.
SeqNumber	int	Sequence number intended to help with temporal ordering. Typically, this will be incremented with every message, but may have gaps depending on the sequencing algorithm being used.
Solicited	boolean	Indicates whether the message is a solicited response to a request or an unsolicited response to an existing event stream. If absent, Solicited defaults to true.
State	object	**Required**. Conveys information about the health of the stream.

Table 18: Refresh Message Structure (Continued)

ATTRIBUTE	TYPE	DEFINITION
└ Code	string,int	Additional status information for the event stream or data state. Not needed for generic state processing. • AlreadyOpen: Indicates that a stream is already open on the connection for the requested data. • AppAuthorizationFailed: Indicates that application authorization using the secure token has failed. • DacsDown: Indicates that the connection to the Data Access Control System is down and users are not allowed to connect. • Error: Indicates an internal error from the sender. • ExceededMaxMountsPerUser: Indicates that the login was rejected because the user exceeded their maximum number of allowed mounts. • FailoverCompleted: Indicated that recovery from a failover condition has finished. • FailoverStarted: Indicates that the component is recovering due to a failover condition. User is notified when recovery finishes via FailoverCompleted. • FullViewProvided: Indicates that the full view (e.g., all available fields) is being provided, even though only a specific view was requested. • GapDetected: Indicates that a gap was detected between messages. • GapFill: Indicates that the received content is meant to fill a recognized gap. • InvalidArgument: Indicates that the request includes an invalid or unrecognized parameter. Specific information should be contained in Text. • InvalidView: Indicates that the requested view is invalid, possibly due to bad formatting. Additional information should be available in Text. • JitConflationStarted: Indicates that JIT conflation has started on the stream. User is notified when JIT conflation ends via RealtimeResume. • MaxLoginsReached: Indicates that the maximum number of logins has been reached. • None: Indicates that additional state code information is not required, nor present. • NotEntitled: Indicates that the request was denied due to permissioning. Typically indicates that the requesting user does not have permission to request on the service, to receive requested data, or to receive data at the requested Quality of Service. • NotFound: Indicates that requested information was not found, though it might be available at a later time or through changing some parameters used in the request. • Preempted: Indicates the stream was preempted, possibly by caching device. Typically indicates the user has exceeded an item limit, whether specific to the user or a component in the system. Relevant information should be contained in Text. • NoBatchViewSupportInReq: Indicates that the provider does not support batch and/or view functionality. • NonUpdatingItem: Indicates that a streaming request was made for non-updating data. • NoResources: Indicates that no resources are available to accommodate the stream. • NotOpen: Indicates that the stream was not opened. Additional information should be available in Text. • RealtimeResumed: Indicated that JIT conflation on the stream has finished. • SourceUnknown: Indicates that the requested service is not known, though the service might be available at a later point in time.

Table 18: Refresh Message Structure (Continued)

ATTRIBUTE	TYPE	DEFINITION
└ Code *(Continued)*	string,int *(Continued)*	• Timeout: Indicates that the timeout occurred somewhere in the system while processing requested data. • TooManyItems: Indicates that a request cannot be processed because too many other streams are already open. • UnableToRequestAsBatch: Indicates that a batch request cannot be used for this request. The user can instead split the batched items into individual requests. • UnsupportedViewType: Indicates that the domain on which a request is made does not support the requests ViewType. • UserAccessToAppDenied: Indicates that the application is denied access to the system. • UsageError: Indicates invalid usage within the system. Specific information should be contained in Text. • UserUnknownToPermSys: Indicates that the user is unknown to the permissioning system and is not allowed to connect.
└ Data	string,int	**Required**. Represents the quality of the data in the response or in the event stream. • NoChange: There is not change in the current state of the data. • Ok: All data associated with the stream is healthy and current. • Suspect: Some or all of the data on a stream is out-of-date (or that it cannot be confirmed as current, e.g., the service is down). If an application does not allow suspect data, a stream might change from Open to Closed or ClosedRecover as a result.
└ Stream	string,int	**Required**. The state of the event stream when using the request/response with interest paradigm. • Closed: Data is not available on this service and connection is not likely to become available, though the data might be available on another service or connection. • ClosedRecover: State is closed, however data can be recovered on this service and connection at a later time. • NonStreaming: The stream is closed and updated data is not delivered without a subsequent re-request. • Open: Data is streaming, as data changes it is sent to the stream. • Redirected: The current stream is closed and has new identifying information. The user can issue a new request for the data using the new message key data from the redirect message.
└ Text	string	Specifies additional information about the current state.
Type	string,int	**Required.** Specifies the message classification. For Refresh messages, Type is Refresh. • Ack • Close • Generic • Post • Refresh • Request • Status • Update

Table 18: Refresh Message Structure (Continued)

27 Request Message

27.1 Request Message Description

A Request message is sent from a consumer to a provider when it wants to request some data, or a capability, available from the provider.

It can also be used to obtain a new response (e.g. synchronization point) or change selected attributes (e.g. priority) for an already open event stream.

27.2 Request Message Structure

ATTRIBUTE	TYPE	DEFINITION
ConfInfoInUpdates	boolean	Specifies whether the consumer wants **ConflationInfo** in updates. ConfInfoInUpdates defaults to false.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketMaker • MarketPrice • MarketByTime • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve
ID	int,array(int)	**Required.** Integer value(s) representing the event stream. It can also be used to match the request and responses.
Key	object	**Required**. The key representing the data content or capability requested.
└ Elements	object	An Element List describing additional attributes of the item stream.

Table 19: Request Message Structure

ATTRIBUTE	TYPE	DEFINITION
Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
Name	string,array(string)	Name(s) of the information requested. If absent, Name defaults to 0.
NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
KeyInUpdates	boolean	Whether the consumer wants the key encoded in every update. If absent, KeyInUpdates defaults to true.
Pause	boolean	Pause item. Pause defaults to false.
Priority	object	When specified, indicates the relative importance of the request and resulting event stream. Priority defaults to Class=1, Count=1.
Class	int	**Required**. Specifies the priority's class.
Count	int	**Required**. Specifies the priority's count.
Private	boolean	Private stream. Private defaults to false.
Qos	object	Specifies the Quality of Service. Provides classification of data/events to provide different tiers of service. • When specified without a WorstQos member, this is the only allowable Quality of Service for the requested stream. If this Quality of Service is unavailable, the stream is not opened. • When specified with a WorstQos, this is the best in the range of allowable Quality of Services. When a Quality of Service range is specified, any Quality of Service within the range is acceptable for servicing the stream. • If neither Qos nor WorstQos are present on the request, this indicates that any available Quality of Service will satisfy the request. • Some components may require Qos on the initial request and on reissue messages. For details, refer to specific component documentation.
Dynamic	boolean	Specifies whether or not the Qos is dynamic. If absent, KeyInUpdates defaults to false.

Table 19: Request Message Structure

ATTRIBUTE	TYPE	DEFINITION
└ Rate	string,int	**Required**. Maximum period of change in data (for streaming events). • JitConflated: Just-in-time conflated, meaning that quality is typically tick-by-tick, but if burst data occurs (or if a component cannot keep up with tick-by-tick delivery), multiple updates are combined into a single update to reduce traffic. This value is usually considered a lower quality than TickByTick. • TickByTick: Tick-by-Tick, meaning data is sent for every update. This is the highest quality of rate value. The best overall Quality of Service is a Rate of TickByTick and a Timeliness of Realtime. • TimeConflated: The interval of time (usually in milliseconds) over which data are conflated is provided in RateInfo. This is a lower quality than TickByTick and at times even lower than JitConflated.
└ RateInfo	int	Specifies any information related to Rate.
└ TimeInfo	int	Information related to Timeliness.
└ Timeliness	string,int	**Required**. Specifies the data's age. • Delayed: Timeliness is delayed and the amount of delay is provided in TimeInfo. This is lower quality than Realtime and might be better than DelayedUnknown. • DelayedUnknown: Timeliness is delayed, although the amount of delay is unknown. This is a lower quality than Realtime and might be worse than Delayed (in which case the delay is known). • Realtime: Data is updated as soon as new data is available, and is the highest-quality Timeliness value. In conjunction with a Rate of TickByTick, Realtime is the best overall Quality of Service.
Qualified	boolean	Qualified stream. Qualified defaults to false.
Refresh	boolean	Indicates whether the user requires a Refresh for this content. This will typically be set to false when changing Priority, View, or when pausing/resuming a stream. Refresh defaults to true.
Streaming	boolean	The application wishes to create an event stream based on this request (i.e. the request/response with interest interaction paradigm). Streaming defaults to true.
Type	string,int	Specifies the message classification. If absent, Type defaults to Request. • Ack • Close • Generic • Post • Refresh • Request • Status • Update
View	array(string,number)	An array of field names or IDs that the client application would like to specifically request.

Table 19: Request Message Structure

ATTRIBUTE	TYPE	DEFINITION
WorstQos	object	Specifies the least acceptable Quality of Service for the requested stream. • When specified with a Qos value, indicates the lower bounds of the quality-of-service required by the application. • When not specified, the best Quality of Service defines the exact Quality of Service required by the application (e.g. application requires realtime/tick-by-tick data).
└ Dynamic	boolean	Whether or not Qos is dynamic. If absent, Dynamic defaults to false.
└ Rate	string,int	**Required**. Specifies the maximum period of change in data (for streaming events). • JitConflated: Just-In-Time Conflated, meaning that quality is typically tick-by-tick, but if burst data occurs (or if a component cannot keep up with tick-by-tick delivery), multiple updates are combined into a single update to reduce traffic. This value is usually considered a lower quality than TickByTick. • TickByTick: Data is sent for every update. This is the highest quality of rate value. The best overall Quality of Service is a Rate of TickByTick and a Timeliness of Realtime. • TimeConflated: The interval of time (usually in milliseconds) over which data are conflated is provided in RateInfo. This is lower quality than TickByTick and at times even lower than JitConflated.
└ RateInfo	int	Information related to Rate.
└ Timeliness	string,int	**Required**. Specifies the data's age. • Delayed: Timeliness is delayed and the amount of delay is provided in TimeInfo. This is lower quality than Realtime and might be better than DelayedUnknown. • DelayedUnknown: Timeliness is delayed, although the amount of delay is unknown. This is a lower quality than Realtime and might be worse than Delayed (in which case the delay is known). • Realtime: Data is updated as soon as new data is available. This is the highest-quality Timeliness value. In conjunction with a Rate of TickByTick, Realtime is the best overall Quality of Service.
└ TimeInfo	int	Information relate to Timeliness.

Table 19: Request Message Structure

28 Status Message

28.1 Status Message Description

The Status message is used to represent asynchronous attribute changes associated with an already opened event stream.

28.2 Status Message Structure

ATTRIBUTE	TYPE	DEFINITION
ClearCache	boolean	An indication that any previous last value payload data cache for the event stream needs to be deleted. If absent, ClearCache defaults to false.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • History • Headline • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • ServiceProviderStatus • Source • Story • SymbolList • System • Reference • ReplayHeadline • ReplayStory • Transaction • YieldCurve
ExtHdr	string	An optional extension to the request message in case an attribute is identified that currently doesn't fit into the request message header.
ID	int	**Required.** Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or requested capability.

Table 20: Status Message Structure

ATTRIBUTE	TYPE	DEFINITION
└ Elements	object	An Element List describing additional attributes of the item stream.
└ Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
└ Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
└ Name	string,array(string)	Name(s) of the information requested.
└ NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
└ Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
PermData	string	Contains permission authorization information for all content provided on this stream.
PostUserInfo	object	Represents information about the posting user.
└ Address	string	**Required**. Dotted-decimal string representing the IP Address of the posting user.
└ UserID	int	**Required**. Specifies the ID of the posting user.
Private	boolean	Private stream. If absent, Private defaults to false.
State	object	Conveys information about the health of the stream.

Table 20: Status Message Structure (Continued)

ATTRIBUTE	TYPE	DEFINITION
└ Code	string,int	Additional status information for the event stream or data state. Not needed for generic state processing. • AlreadyOpen: Indicates that a stream is already open on the connection for the requested data. • AppAuthorizationFailed: Indicates that application authorization using the secure token has failed. • DacsDown: Indicates that the connection to Data Access Control System is down and users are not allowed to connect. • Error: Indicates an internal error from the sender. • ExceededMaxMountsPerUser: Indicates that the login was rejected because the user exceeded their maximum number of allowed mounts. • FailoverCompleted: Indicated that recovery from a failover condition has finished. • FailoverStarted: Indicates that the component is recovering due to a failover condition. User is notified when recovery finishes via FailoverCompleted. • FullViewProvided: Indicates that the full view (e.g., all available fields) is being provided, even though only a specific view was requested. • GapDetected: Indicates that a gap was detected between messages. • GapFill: Indicates that the received content is meant to fill a recognized gap. • InvalidArgument: Indicates that the request includes an invalid or unrecognized parameter. Specific information should be contained in Text. • InvalidView: Indicates that the requested view is invalid, possibly due to bad formatting. Additional information should be available in Text. • JitConflationStarted: Indicates that JIT conflation has started on the stream. User is notified when JIT conflation ends via RealtimeResume. • MaxLoginsReached: Indicates that the maximum number of logins has been reached. • NoBatchViewSupportInReq: Indicates that the provider does not support batch and/or view functionality. • None: Indicates that additional state code information is not required, nor present. • NonUpdatingItem: Indicates that a streaming request was made for non-updating data. • NoResources: Indicates that no resources are available to accommodate the stream. • NotEntitled: Indicates that the request was denied due to permissioning. Typically indicates that the requesting user does not have permission to request on the service, to receive requested data, or to receive data at the requested Quality of Service. • NotFound: Indicates that requested information was not found, though it might be available at a later time or through changing some parameters used in the request. • NotOpen: Indicates that the stream was not opened. Additional information should be available in Text. • Preempted: Indicates the stream was preempted, possibly by caching device. Typically indicates the user has exceeded an item limit, whether specific to the user or a component in the system. Relevant information should be contained in Text. • RealtimeResumed: Indicated that JIT conflation on the stream has finished. • SourceUnknown: Indicates that the requested service is not known, though the service might be available at a later point in time. • Timeout: Indicates that the timeout occurred somewhere in the system while processing requested data.

ATTRIBUTE	TYPE	DEFINITION
└ Code *(continued)*	string,int *(continued)*	• TooManyItems: Indicates that a request cannot be processed because too many other streams are already open. • UnableToRequestAsBatch: Indicates that a batch request cannot be used for this request. The user can instead split the batched items into individual requests. • UnsupportedViewType: Indicates that the domain on which a request is made does not support the requests ViewType. • UsageError: Indicates invalid usage within the system. Specific information should be contained in Text. • UserAccessToAppDenied: Indicates that the application is denied access to the system. • UserUnknownToPermSys: Indicates that the user is unknown to the permissioning system and is not allowed to connect.
└ Data	string,int	Represents the quality of the data in the response or in the event stream. • NoChange: There is not change in the current state of the data. • Ok: All data associated with the stream is healthy and current. • Suspect: Some or all of the data on a stream is out-of-date (or that it cannot be confirmed as current, e.g., the service is down). If an application does not allow suspect data, a stream might change from Open to Closed or ClosedRecover as a result.
└ Stream	string,int	The state of the event stream when using the request/response with interest paradigm. • Closed: Data is not available on this service and connection is not likely to become available, though the data might be available on another service or connection. • ClosedRecover: State is closed, however data can be recovered on this service and connection at a later time. • NonStreaming: The stream is closed and updated data is not delivered without a subsequent re-request. • Open: Data is streaming, as data changes it is sent to the stream. • Redirected: The current stream is closed and has new identifying information. The user can issue a new request for the data using the new message key data from the redirect message.
└ Text	string	Specifies additional information about the current state.
Type	string,int	**Required.** Specifies the message classification. For Status messages, Type is Status. • Ack • Close • Generic • Post • Refresh • Request • Status • Update
Qualified	boolean	Qualified stream. If absent, Qualified defaults to false.

Table 20: Status Message Structure (Continued)

29 Update Message

29.1 Update Message Description

The Update message is used to represent asynchronous payload data events associated with an already opened event stream.

Domain models may assign different meaning to Updates depending on the actual content modelled.

29.2 Update Message Structure

ATTRIBUTE	TYPE	DEFINITION
ConflationInfo	object	When requested, provides the information about any conflation logic that may have been applied to this event.
└ Count	int	**Required**. Conflation count.
└ Time	int	**Required**. Conflation time.
Discardable	boolean	Specifies whether the message can be discarded.If absent, Discardable defaults to false.
Domain	string,int	Specifies the domain model represented by this message. If absent, Domain defaults to MarketPrice. • Analytics • Contribution • Dictionary • EconomicIndicator • Forecast • Headline • History • Login • MarketByOrder • MarketByPrice • MarketByTime • MarketMaker • MarketPrice • NewsTextAnalytics • Poll • ProviderAdmin • Reference • ReplayHeadline • ReplayStory • ServiceProviderStatus • Source • Story • SymbolList • System • Transaction • YieldCurve

Table 21: Update Message Structure

ATTRIBUTE	TYPE	DEFINITION
DoNotCache	boolean	Specifies whether to apply this update to cache. If absent, DoNotCache defaults to false.
DoNotConflate	boolean	Specifies whether to conflate payload data in this particular update. If absent, DoNotConflate defaults to false.
DoNotRipple	boolean	Specifies whether to ripple fields in the update. If absent, DoNotRipple defaults to false.
ExtHdr	string	An optional extension to the request message in case an attribute is identified that currently doesn't fit into the request message header.
ID	int,array(int)	**Required.** Integer value representing the event stream. It can also be used to match the request and responses.
Key	object	The key representing the data content or capability requested.
└ Elements	object	An Element List describing additional attributes of the item stream.
└ Filter	object	A filter specification used to request which filter entries will be present in a Filter List payload.
└ Identifier	int	A user-defined numeric identifier. Identifier is defined on a per-domain basis. This attribute's range is from -2,147,483,648 to 2,147,483,647.
└ Name	string,array(string)	Name(s) of the information requested.
└ NameType	string,int	An enumeration representing the different forms the name can take. If absent, NameType defaults to Ric. • AuthnToken: Authentication Token • Cookie: User information is specified in cookie • EmailAddress: Email Address • Name: Username • Ric: Reuters Instrument Code • Token: User Token (Typically AAA Token) • Unspecified: Unspecified
└ Service	string,int	A name or ID representing the identifier of the service provider. If absent, Service defaults to the default service in the LSEG Real-Time Advanced Distribution Server configuration.
PermData	string	Contains permission authorization information for all content provided on this stream.
PostUserInfo	object	Represents information about the posting user.
└ Address	string	**Required**. Dotted-decimal string representing the IP Address of the posting user.
└ UserID	int	**Required**. Specifies the ID of the posting user.
SeqNumber	int	Sequence number intended to help with temporal ordering. Typically, this will be incremented with every message, but may have gaps depending on the sequencing algorithm being used.

Table 21: Update Message Structure (Continued)

ATTRIBUTE	TYPE	DEFINITION
Type	string,int	**Required.** Specifies the message classification. For Update messages, Type is Update. • Ack • Close • Generic • Post • Refresh • Request • Status • Update
UpdateType	string,int	Specifies the type of update as defined by the domain model. If absent, UpdateType defaults to Unspecified. • ClosingRun: Closing run. • Correction: Correction. • MarketDigest: Market digest. • Multiple: Update event with filtering and conflation applied. • NewsAlert: News alert. • OrderIndication: Order indication. • Quote: Quote. • QuotesTrade: Quotes followed by a Trade. • Trade: Trade. • Unspecified: Unspecified update event. • Verify: Fields may have changed. • VolumeAlert: Volume alert.

Table 21: Update Message Structure (Continued)

30 Domain Model Usage: Market Price Domain

30.1 Market Price Domain Overview

The ***Market Price*** domain provides access to Level I market information such as trades, indicative quotes, and top-of-book quotes. All information is sent as a **`FieldList`**. Field-value pairs contained in the field list include information related to that item (i.e., net change, bid, ask, volume, high, low, or last price).

NOTE: **`GenericMsg`**(s) are not supported in the **MarketPrice** Domain Model.

Refer to the following topics for details on Market Price domain message types:

- Usage: Market Price Request Message
- Usage: Market Price Refresh Message
- Usage: Market Price Update Message
- Usage: Market Price Status Message

30.2 Market Price Domain Examples

The following message samples illustrate the use of the Market Price Domain.

30.2.1 Market Price Request Message Sent

```
{
    "ID": 2,
    "Key": {
        "Name": "TRI.N"
    }
}
```

30.2.2 Market Price Refresh Message Received

```
[
    {
        "ID": 2,
        "Type": "Refresh",
        "Key": {
            "Service": "ELEKTRON_DD",
            "Name": "TRI.N"
        },
        "State": {
            "Stream": "Open",
            "Data": "Ok",
            "Text": "All is well"
        },
        "Qos": {
            "Timeliness": "Realtime",
```

```
            "Rate": "TimeConflated",
            "RateInfo": 1000
        },
        "PermData": "AwO9ZWLA",
        "SeqNumber": 56256,
        "Fields": {
            "PROD_PERM": 6562,
            "RDNDISPLAY": 64,
            "DSPLY_NAME": "THOMSON REUTERS",
            "RDN_EXCHID": "NYS",
            "TRDPRC_1": 39.71,
            "TRDPRC_2": 39.71,
            "TRDPRC_3": 39.7,
            "TRDPRC_4": 39.71,
            "TRDPRC_5": 39.7,
            "NETCHNG_1": -0.16,
            "HIGH_1": 39.82,
            "LOW_1": 39.48,
            "PRCTCK_1": "?",
            "CURRENCY": "USD",
            "TRADE_DATE": "2018-04-06",
            "TRDTIM_1": "15:37:00",
            "OPEN_PRC": 39.5,
            "HST_CLOSE": 39.87,
            "BID": 39.7,
            "BID_1": 39.7,
            "BID_2": 39.7,
            "ASK": 39.72,
            "ASK_1": 39.72,
            "ASK_2": 39.72,
            "NEWS": "YYYY",
            "NEWS_TIME": "10:57:36",
            "BIDSIZE": 8,
            "ASKSIZE": 8,
            "ACVOL_1": 115480,
            "EARNINGS": 1.5162,
            "YIELD": 3.4612,
            "PERATIO": 26.2965,
            "DIVIDENDTP": "  ",
            "DIVPAYDATE": "2018-03-15",
            "EXDIVDATE": "2018-02-21",
            "CTS_QUAL": "MSW",
            "BLKCOUNT": 1,
            "BLKVOLUM": 20479,
            "TRD_UNITS": "6DP ",
            "PCTCHNG": -0.4013,
            "DJTIME": null,
            "CLOSE_BID": 39.87,
            "CLOSE_ASK": 39.88,
            "DIVIDEND": 1.38,
```

```
            "UPLIMIT": 43.45,
            "LOLIMIT": 35.55,
            "NUM_MOVES": 636,
            "OFFCL_CODE": "000884903105",
            "HSTCLSDATE": "2018-04-05",
            "YRHIGH": 48.6,
            "YRLOW": 38.22,
            "TURNOVER": null,
            "BOND_TYPE": null,
            "BCKGRNDPAG": null,
            "YCHIGH_IND": null,
            "YCLOW_IND": null,
            "CUM_EX_MKR": "    ",
            "PRC_QL_CD": "R  ",
            "PRC_QL2": "    ",
            "TRDVOL_1": 100,
            "LOT_SIZE_A": 100,
            "RECORDTYPE": 113,
            "BID_MMID1": null,
            "ASK_MMID1": null,
            "OPTION_XID": "PABCEH",
            "YRHIGHDAT": "2017-10-17",
            "YRLOWDAT": "2018-03-28",
            "IRGPRC": 39.7,
            "IRGVOL": 20,
            "IRGCOND": "ODD",
            "TIMCOR": null,
            "INSPRC": null,
            "INSVOL": null,
            "INSCOND": null,
            "SALTIM": "15:37:14",
            "BCAST_REF": "TRI.TO",
            "OFF_CD_IND": "CUS",
            "GEN_VAL3": 43.45,
            "GEN_VAL4": 35.55,
            "GV1_TEXT": " ",
            "GV2_TEXT": "X",
            "GV3_TEXT": " ",
            "GV4_TEXT": "    I",
            "SEQNUM": 498472,
            "PRNTYP": " ",
            "PRNTBCK": null,
            "QUOTIM": "15:37:31",
            "GV1_FLAG": " ",
            "GV2_FLAG": " ",
            "GV3_FLAG": " ",
            "GV4_FLAG": " ",
            "OFF_CD_IN2": null,
            "OFFC_CODE2": "MKNPTzBegXCA",
            "EXCHTIM": "15:37:14",
```

```
            "YRHI_IND": "Yr.High ",
            "YRLO_IND": "Yr.Low  ",
            "PREF_DISP": 5752,
            "VOL_X_PRC1": 39.6576,
            "DSO_ID": null,
            "CLOSE_TIME": null,
            "ODD_VOLUME": 3407,
            "ADJUST_CLS": 39.87,
            "STOCK_TYPE": "A",
            "IMP_VOLT": null,
            "RDN_EXCHD2": "NYS",
            "YEAR_FCAST": "08I",
            "IRGVAL": 4,
            "LIST_MKT": "N",
            "PCT_ABNVOL": 0.4361,
            "BC_10_50K": 1,
            "BC_50_100K": null,
            "BC_100K": null,
            "PMA_50D": 40.285,
            "PMA_150D": 43.4143,
            "PMA_200D": 44.1181,
            "VMA_10D": 264812,
            "VMA_25D": 238154,
            "VMA_50D": 339041,
            "OPN_NETCH": -0.37,
            "PREV_DISP": 0,
            "PRC_QL3": "R  ",
            "52WK_HIGH": 48.6,
            "52WK_LOW": 38.22,
            "MPV": "INT ",
            "OFF_CLOSE": null,
            "QUOTE_DATE": "2018-04-06",
            "VWAP": 39.6576,
            "PROV_SYMB": "TRI",
            "52W_HDAT": "2017-10-17",
            "52W_HIND": null,
            "52W_LDAT": "2018-03-28",
            "52W_LIND": null,
            "BID_ASK_DT": "2018-04-05",
            "CRSTRD_PRC": null,
            "MNEMONIC": "TRI",
            "LOLIMIT_2": null,
            "UPLIMIT_2": null,
            "PERIOD_CDE": null,
            "TRDTIM_MS": 56234366,
            "SALTIM_MS": 56234366,
            "QUOTIM_MS": 56251678,
            "TIMCOR_MS": null,
            "BLK_PRC1": 39.5,
            "OPN_AUCVOL": null,
```

```
            "CLS_AUCVOL": null,
            "PDTRDPRC": null,
            "PREDAYVOL": null,
            "PDTRDDATE": null,
            "SEQNUM_QT": 12463543,
            "FIN_STATUS": "N",
            "LS_SUBIND": " ",
            "IRG_SUBIND": " ",
            "TRADE_ID": "53017968850743",
            "MKT_STATUS": null,
            "TRD_TYPE": null,
            "IPO_PRC": null,
            "ODD_PRC": 39.7,
            "RCS_AS_CLA": " ",
            "IMB_ACT_TP": null,
            "IMB_SH": null,
            "IMB_SIDE": null,
            "IMB_TIM_MS": null,
            "TRD_THRU_X": "X",
            "IRG_TDTH_X": " ",
            "IRGDATE": "2018-04-06",
            "TURN_BLOCK": 808920.5,
            "DOM_EQ_ID": "MKNPTzBegXCA",
            "DOM_OPT_ID": "PWXYZ",
            "CUSIP_CD": "884903105",
            "LSTSALCOND": " F  ",
            "IRGSALCOND": "   I",
            "INSSALCOND": null,
            "THRESH_IND": "1",
            "CANCEL_IND": null,
            "RETRAN_IND": "1",
            "CONTEXT_ID": 1070,
            "IRG_TRDID": "53017902921778",
            "PRC_TICK": 0.01,
            "VWAP_VOL": 115480,
            "POST_PANEL": "08I",
            "IRG_SEQNO": 492570,
            "INS_SEQNO": null,
            "OFF_CL_TIM": null,
            "DDS_DSO_ID": 8357,
            "SPS_SP_RIC": ".[SPS23SNJL1",
            "SETL_TYPE": "NRM",
            "BOOK_STATE": "N",
            "HALT_REASN": null,
            "SH_SAL_RES": "N",
            "BID_COND_N": "R",
            "ASK_COND_N": "R",
            "CAN_PRC": null,
            "CAN_VOL": null,
            "CAN_COND": null,
```

```
            "CAN_COND_N": null,
            "CAN_TRD_ID": null,
            "REPORT_VOL": 115480,
            "TRD_STATUS": null,
            "HALT_RSN": "  ",
            "CTRDTIM_MS": null,
            "CTRDTIM": null,
            "INSTRD_TIM": null,
            "OFF_CLS_DT": null,
            "CAN_DATE": null,
            "INSTRD_DT": null,
            "PD_SEQNO": null,
            "BLKTRDVOL": 20479,
            "PDACVOL": null,
            "AC_VOL_CRS": 0,
            "ODD_TRDVOL": 20,
            "CAN_SEQNO": null,
            "ELG_NUMMOV": 514,
            "BLK_SEQNO": 2407,
            "ODD_SEQNO": 492570,
            "CRS_SEQNO": null,
            "CRS_TRDVOL": null,
            "CRS_NUMOV": null,
            "AC_TRN_CRS": null,
            "SEE_RIC": null,
            "BCASTREF32": null,
            "QTE_ORIGIN": " ",
            "CAN_TDTH_X": null,
            "CAN_SUBIND": null,
            "PD_TDTH_X": null,
            "PD_SUBIND": null,
            "INS_TDTH_X": null,
            "INS_SUBIND": null,
            "XMIC_CODE": "XNYS",
            "CRSSALCOND": null,
            "PD_SALCOND": null,
            "RCS_AS_CL2": null,
            "PD_TRDID": null,
            "PERIOD_CD2": null,
            "INS_TRDID": null,
            "BLK_TRDID": "52983642797361",
            "CRS_TRDID": null,
            "ODD_TRDID": "53017902921778",
            "REG_PILOT": null,
            "INST_PHASE": null,
            "RETAIL_INT": "B  ",
            "LIMIT_IND2": "LMT",
            "LIMIT_INDQ": "   ",
            "MK_STATUS": null,
            "LULD_TM_MS": 56250021,
```

```
            "INSTIM_MS": null,
            "IRGTIM_MS": 56119022,
            "PRE_013_MS": null,
            "BLK_DATE": "2018-04-06",
            "CRS_DATE": null,
            "ODD_DATE": "2018-04-06",
            "LMT_REFPR2": null,
            "PRERL1348": null,
            "ODDSALCOND": "   I",
            "BLKSALCOND": " O  ",
            "SECUR_ST": "F",
             "DTRS_IND": null,
             "IVOC_IND": null,
             "BLK_FLAG": null,
             "VWAP_FLAG": null,
             "CAN_TERMS": null,
             "NBBO_IND": "4  ",
             "TEST_MSG": null,
             "STATUS_IND": null,
             "HELD_T_IND": "   ",
             "PRE_2ET262": null,
             "PRE_2ET263": null,
             "BLKTIM_MS": "13:30:00.908",
             "PDTRDTM_MS": null,
             "ORDRECV_MS": "15:37:31.678",
             "TRDRECV_MS": "15:37:14.366",
             "ORDREC2_MS": null,
             "TRDREC2_MS": null,
             "CRSTIM_MS": null,
             "ODDTIM_MS": "15:35:19.022",
             "HALT_TM_MS": null,
             "OFF_CLS_MS": null,
             "CNTX_VER_N": null,
             "DM_TYPE": null,
             "ELG_ACVOL": 112073,
             "ELG_TNOV": 4444393.91,
             "ODDTRN_UNS": 135268.085,
             "TRNOVR_UNS": 4579661.995,
             "ACVOL_UNS": 115480
        }
    }
]
```

30.2.3 Market Price Update Message Received


```
[
    {
        "ID": 2,
        "Type": "Update",
        "UpdateType": "Unspecified",
        "Key": {
            "Service": "ELEKTRON_DD",
            "Name": "TRI.N"
        },
        "SeqNumber": 56352,
        "Fields": {
            "BID": 39.7,
            "ASK": 39.72,
            "BIDSIZE": 8,
            "ASKSIZE": 8,
            "BID_COND_N": "R",
            "ASK_COND_N": "R",
            "GV1_TEXT": " ",
            "LIMIT_INDQ": "    ",
            "PRC_QL_CD": "R  ",
            "PRC_QL3": "R  ",
            "QTE_ORIGIN": " ",
            "GV1_FLAG": " ",
            "SEQNUM_QT": 12465483,
            "RETAIL_INT": "B  ",
            "STOCK_TYPE": "A",
            "NBBO_IND": "4  ",
            "QUOTIM_MS": 56254624,
            "QUOTIM": "15:37:34",
            "SETL_TYPE": "NRM",
            "BOOK_STATE": "N",
            "ORDRECV_MS": "15:37:34.624"
        }
    }
]
```

30.3 Usage: Market Price Request Message

A Market Price request message is encoded using **RequestMsg** and sent by Open Message Model consumer applications. The request specifies the name and attributes of an item in which the consumer is interested.

To receive updates, a consumer can make a "streaming" request by setting **Request.Streaming** to **true**. If the flag is not set, the consumer requests a "snapshot," and the refresh ends the request (though updates might be received in either case if the refresh has multiple parts).

To stop updates, a consumer can pause an item (if the provider supports the pause feature). For additional details, refer to the *Enterprise Transport API Java Edition Developers Guide*.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketPrice 6**
Interactions	**Required**. • **InitialImage**: **true**, indicates that an initial image is required. • **InterestAfterRefresh**: **true**, indicates that a streaming request is required. • **Pause**: **true**, indicates that a pause is required.
Indications	Optional. **ConflatedInUpdates**: **true**, indicates conflated updates is required Batch and View request are specified in the **Payload**.
Qos	Optional. Indicates the QoS at which the consumer wants the stream serviced. If both **Qos** and **WorstQos** are specified, this request can be satisfied by a range of QoS.
WorstQos	Optional. Used with the **Qos** member to define a range of acceptable QoS. When the provider encounters such a range, it should attempt to provide the best QoS it can within that range. **WorstQos** should only be used on services that claim to support it via the **SupportsQosRange** item in the Source Directory response (refer to).
	NOTE: Enterprise Message API provides the **Request.Qos()** method to set both **Qos** and **WorstQos** depending upon the timeliness and rate values.
Priority	Optional. Indicates the class and count associated with stream priority.
ExtHdr	Not used.
Key.Service	**Required**. Specifies the ID of the service from which the consumer wishes to request the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. When consuming from LSEG sources, typically set to **Ric** (the "Instrument Code"). If unspecified, **Key.NameType** defaults to **Ric**.
Key.Name	**Required**. Specifies the name of the requested item.
ServiceName	**Required**. Specifies the name of the service from which the consumer wishes to request the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Optional. When features such as View are leveraged, the payload can contain information relevant to that feature. For more detailed information, refer to the Appendix A.

Table 22: Market Price Request Message

30.4 Usage: Market Price Refresh Message

A Market Price Refresh Message is encoded using **RefreshMsg** and sent by Open Message Model provider and non-interactive provider applications. This message sends all currently available information about the item to the consumer.

FieldList in the payload should include all fields that may be present in subsequent updates, even if those fields are currently blank. When responding to a View request, this refresh should contain all fields that were requested by the specified view. If for any reason the provider wishes to send new fields, it must first send an unsolicited refresh with both the new and currently-present fields.

NOTE: All solicited or unsolicited refresh messages in the Market Price domain must be atomic. The Market Price domain does not allow for multi-part refresh use. The provider should only send the **Key.Name** and **ServiceName** in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true** in the Enterprise Message API configuration, then the **Key.Name** and **ServiceName** must be provided for every Refresh response messages.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketPrice**
State	**Required**. Includes the state of the stream and data.
Solicited	**Required**. Indicates whether the refresh was solicited. Possible settings are: • **true**: The refresh was solicited. • **false**: The refresh was unsolicited.
Indications	**Required.** Available settings include: • **Complete**: **true**, Indicates that the refresh is complete. • **DoNotCache**: **true**, Indicates that the refresh message should not be cached. • **StatusMsgFlags.Clear_Cache**: **true**, Indicates to clear the cache.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	Optional.Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies the permission information associated with content on this stream.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service from which the consumer wishes to request the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Required**. Specifies the ID of the service that provides the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. **Key.NameType** should match the **Key.NameType** specified in the request. If unspecified, **Key.NameType** defaults to **Ric**.
Key.Name	**Required**. This should match the requested name.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	**Required**. This should consist of a **FieldList** containing all fields associated with the item.

Table 23: Market Price Refresh Message

30.5 Usage: Market Price Update Message

A Market Price Update Message is encoded using **UpdateMsg** and sent by Open Message Model provider and non-interactive provider applications. The Market Price Update Message conveys any changes to an item's data.

NOTE: The provider should only send the **Key.Name** and **Key.NameType** in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **Key.Name** and **Key.NameType** must be provided for every Update response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketPrice 6**
UpdateType	**Required**. Indicates the general content of the update: • **Unspecified 0** • **Quote 1** • **Trade 2** • **NewsAlert 3** • **VolumeAlert 4** • **OrderIndication 5** • **ClosingRun 6** • **Correction 7** • **MarketDigest 8** • **QuotesTrade 9** • **Multiple 10** • **Verify 11**
Indications	**Conditional.** • If **UpdateType** is set to be **Correction=7** or **INSTRUMENT_UPDATE_VERIFY=11**, **DoNotRipple** must be set to **true**. • **DoNotCache**: **true**, Indicates the application should not cache this update message. • **DoNotConflate**: **true**, Indicates the application should not conflate updates.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
ConflationInfo.Count	Optional. If a provider sends a conflated update, **ConflationInfo.Count** specifies the number of updates in the conflation. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** to **true** in the request.
ConflationInfo.Time	Optional. If a provider sends a conflated update, **ConflationInfo.Time** specifies the time interval (in milliseconds) over which data is conflated. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** to **true** in the request.
GroupId	Optional. Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies permissioning information associated with only the contents of this update.
ExtHdr	Not used.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true** on the request. Specifies the ID of the service that provides the data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.

Table 24: Market Price Update Message

COMPONENT	DESCRIPTION / VALUE
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true** on the request. **Key.NameType** should match the name type specified on the request. If **Key.NameType** is unspecified, its value defaults to **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true** on the request. **Key.Name** specifies the name of the item being provided.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the service that provides the data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	**Required**. This should consist of a **FieldList** with any changed data.

Table 24: Market Price Update Message (Continued)

30.6 Usage: Market Price Status Message

A Market Price Status Message is encoded using **StatusMsg** and sent by Open Message Model provider and non-interactive provider applications. The status message conveys state change information associated with an item stream.

NOTE: The provider should only send the **Key.Name** and **Key.NameType** in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **Key.Name** and **Key.NameType** must be provided for every Status response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketPrice 6**
State	Optional. Specifies the current state information associated with the data and stream.
GroupId	Optional. Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies permissioning information associated with only the contents of this message.
ExtHdr	Not used.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true** on the request. Specifies the ID of the service that provides the data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true** on the request. **Key.NameType** should match the name type specified on the request. If **Key.NameType** is unspecified, its value defaults to **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true** on the request. **Key.Name** specifies the name of the item being provided.

Table 25: Market Price Status Message

COMPONENT	DESCRIPTION / VALUE
ServiceName	**Conditional**. **`ServiceName`** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the service that provides the data.
	NOTE: The application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Not used.

Table 25: Market Price Status Message (Continued)

31 Domain Model Usage: Market by Price Domain

31.1 Market by Price Domain Overview

Market By Price provides access to Level II market depth information. The list of price points is sent in a **`Map`**. Each entry represents one price point (using that price and bid/ask side as its key) and contains a **`FieldList`** that describes information related to that price point.

NOTE: **`GenericMsg`**(s) are not supported for the **MarketByPrice** Domain Model.

Refer to the following topics for details on Market by Price domain message types:

- Usage: Market by Price Request Message
- Usage: Market by Price Refresh Message
- Usage: Market by Price Update Message
- Usage: Market by Price Status Message

31.2 Market by Price Domain Examples

The following message samples illustrate the use of the Market by Price Domain.

31.2.1 Market by Price Request Message Sent

```
{
    "ID": 2,
    "Domain": "MarketByPrice",
    "Key": {
        "Name": "TRI.TO"
    }
}
```

31.2.2 Market by Price Refresh Message Received

```
[
    {
        "ID": 2,
        "Type": "Refresh",
        "Domain": "MarketByPrice",
        "Key": {
            "Service": "ELEKTRON_DD",
            "Name": "TRI.TO"
        },
        "State": {
            "Stream": "Open",
            "Data": "Ok",
            "Text": "All is well"
        },
        "Qos": {
```

```
            "Timeliness": "Realtime",
            "Rate": "TimeConflated",
            "RateInfo": 1000
        },
        "PermData": "AwO9META",
        "SeqNumber": 63888,
        "Map": {
            "KeyType": "Buffer",
            "Summary": {
                "Fields": {
                    "PROD_PERM": 3044,
                    "DSPLY_NAME": "THOMSON REUTERS",
                    "CURRENCY": "CAD",
                    "ACTIV_DATE": "2018-04-06",
                    "PRC_QL2": " AU",
                    "LOT_SIZE_A": 100,
                    "RECORDTYPE": 113,
                    "PREF_DISP": null,
                    "RDN_EXCHD2": "TOR",
                    "LIST_MKT": "TOR",
                    "PROV_SYMB": "TRI",
                    "PR_RNK_RUL": "NOR",
                    "OR_RNK_RUL": "PTS ",
                    "MNEMONIC": "TRI",
                    "MKT_STATUS": "S",
                    "TIMACT_MS": 67206707,
                    "CONTEXT_ID": 2171,
                    "DDS_DSO_ID": 8244,
                    "SPS_SP_RIC": ".[SPSTL2TRL2",
                    "BOOK_STATE": "N",
                    "HALT_REASN": null,
                    "ORD_ENT_ST": "E",
                    "MKT_OR_RUL": " ",
                    "TRD_STATUS": "N ",
                    "HALT_RSN": null
                }
            },
            "CountHint": 148,
            "Entries": [
                {
                    "Action": "Add",
                    "Key": "NTAuNTEwMDAwQQ==",
                    "Fields": {
                        "ORDER_PRC": 50.51,
                        "ORDER_SIDE": "ASK",
                        "NO_ORD": 1,
                        "ACC_SIZE": 100,
                        "LV_TIM_MS": 67200951,
                        "LV_TIM_MSP": 471,
                        "LV_DATE": "2018-04-06"
```

```
                    }
                },
                ...
                {
                    "Action": "Add",
                    "Key": "NjEuMDAwMDAwQQ==",
                    "Fields": {
                        "ORDER_PRC": 61,
                        "ORDER_SIDE": "ASK",
                        "NO_ORD": 2,
                        "ACC_SIZE": 200,
                        "LV_TIM_MS": 44172101,
                        "LV_TIM_MSP": 151,
                        "LV_DATE": "2018-03-19"
                    }
                }
            ]
        }
    }
]
```

31.2.3 Market by Price Update Message Received

```
[
    {
        "ID": 2,
        "Type": "Update",
        "Domain": "MarketByPrice",
        "UpdateType": "Unspecified",
        "Key": {
            "Service": "ELEKTRON_DD",
            "Name": "TRI.TO"
        },
        "SeqNumber": 63904,
        "Map": {
            "KeyType": "Buffer",
            "Summary": {
                "Fields": {
                    "TIMACT_MS": 67207819
                }
            },
            "Entries": [
                {
                    "Action": "Update",
                    "Key": "NTAuNDkwMDAwQg==",
                    "Fields": {
                        "ORDER_PRC": 50.49,
                        "ORDER_SIDE": "BID",
```

```
                        "ACC_SIZE": 700,
                        "NO_ORD": 6,
                        "LV_TIM_MS": 67206707,
                        "LV_TIM_MSP": 488,
                        "LV_DATE": "2018-04-06"
                    }
                }
            ]
        }
    }
]
```

Usage: Market by Price Request Message

A Market By Price request message is encoded using **RequestMsg** and sent by Open Message Model consumer applications. The request specifies the name of an item in which the consumer is interested.

To receive updates, a consumer can make a "streaming" request by setting **Request.Streaming** to **true**. If the flag is not set, the consumer requests a "snapshot" and the refresh should end the request (updates may be received in either case if the refresh has multiple parts).

A consumer can pause an item to stop updates (if the provider supports such functionality). For more information, refer to the *Enterprise Transport API Java Edition Developers Guide*.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByPrice 8**
Interactions	**Conditional**. • **InitialImage**: **true**, indicates that an initial image is required. • **InterestAfterRefresh**: **true**, indicates that a streaming request is required. • **Pause**: **true**, indicates that a pause is required.
Indications	Optional. **ConflatedInUpdates**: **true**, indicates that conflated updates are required. Batch and View requests are specified in the Payload.
Qos	Optional. Indicates the QoS at which the consumer wants the stream serviced. If both **Qos** and **WorstQos** are specified, this request can be satisfied by a range of QoS.
WorstQos	Optional. Used with **Qos** to define a range of acceptable QoS. When the provider encounters such a range, it should attempt to provide the best QoS possible within that range. This should only be used on services that claim to support it via the **SupportsQosRange** item in the Source Directory response (for further details, refer to Section 4.3.1.1).
	NOTE: Enterprise Message API provides the **Request.Qos()** method to set both **Qos** and **WorstQos** depending upon the timeliness and rate values.
Priority	Optional. Indicates the class and count associated with stream priority.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service from which the consumer wishes to request data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.

Table 26: Market By Price Request Message

COMPONENT	DESCRIPTION / VALUE
Key.Service	**Required**. Specifies the ID of the service that provides the requested item.
	NOTE: The application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.NameType	Optional. Typically set to **Ric** (the "Instrument Code") when consuming from LSEG sources. If absent, its default value is **Ric**.
Key.Name	**Required**. Specifies the name of the requested item.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Optional. When features such as View or Batch are leveraged, the payload can contain information relevant to that feature. For further details, refer to the Appendix A.

Table 26: Market By Price Request Message (Continued)

31.3 Usage: Market by Price Refresh Message

A Market By Price refresh message is encoded using **RefreshMsg** and sent by Open Message Model interactive provider and non-interactive provider applications.

A Market By Price refresh may be sent in multiple parts. Both update and status messages can be delivered between parts of a refresh message, regardless of streaming or non-streaming request.

NOTE: The provider should send **Key.Name** and **ServiceName** only in the first Refresh response message, unless **MsgKeyInUpdates** is set to **true** in the Enterprise Message API configuration. In this case **Key.Name** and **ServiceName** must be provided in each Refresh response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByPrice 8**
State	**Required**. Indicates the state of the stream and data.
Solicited	**Required**. Indicates whether the refresh message was solicited. • **true**: The refresh message is solicited. • **false**: The refresh message is unsolicited.
Indications	**Conditional**. • **DoNotCache**: **true**, indicates that the application should not cache this refresh message. • **StatusMsgFlags.Clear_Cache**: **true**, indicates that the application should clear its cache. • **Complete**: **true**, indicates that this is the last message in the refresh complete.
PartNum	Optional. Specifies the part number of a multi-part refresh.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	Optional. Associates the item with an Item Group (for further information, refer to Section 4.3.1.3).
PermData	Optional. If present, specifies permission information associated with the stream's content.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service that provides the item. **NOTE:** The consumer application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Required**. Specifies the ID of the service that provides the item. **NOTE:** The consumer application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. **Key.NameType** should match the **Key.NameType** specified in the request. If absent, this value is assumed to be **Ric**.
Key.Name	**Required**. **Key.Name** should match the name specified in the request.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.

Table 27: Market By Price Refresh Message

COMPONENT	DESCRIPTION / VALUE
Payload	**Required**. The order book is represented by a **`Map`**, where each entry (**`MapEntry`**) contains a **`FieldList`** which has information about a price point.

Table 27: Market By Price Refresh Message (Continued)

31.4 Usage: Market by Price Update Message

A Market By Price update message is encoded using **UpdateMsg** and sent by Open Message Model interactive provider and non-interactive provider applications. The provider can send an update message to add, update, or remove price point information. Updates will not be received before images. True snapshots are supported.

NOTE: The provider should send **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then **Key.Name** and **ServiceName** must be provided for every Update response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByPrice 8**
UpdateType	**Required**. Indicates the general content of the update. Typically sent as one of the following: • **Unspecified 0** • **Quote 1**
Indications	Optional. • **DoNotCache**: **true**, indicates to not cache the update message. • **DoNotConflate**: **true**, indicates to not conflate the update message.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
ConflationInfo.Count	Optional. If a provider sends a conflated update, **ConflationInfo.Count** specifies how many updates were included in the conflation. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** flag in the request.
ConflationInfo.Time	Optional. If a provider sends a conflated update, **ConflationInfo.Time** specifies the time interval (in milliseconds) over which data is conflated. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** flag in the request.
PermData	Optional. Specifies permissioning information for the update's content.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the service that provides the data. **NOTE:** The provider application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set on the request. Specifies the ID of the service that provides the item. **NOTE:** The provider application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true** on the request. **Key.NameType** should match the **Key.NameType** specified in the item's request message. If **Key.NameType** is not specified, it uses the default **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true** on the request) Specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.

Table 28: Market By Price Update Message

COMPONENT	DESCRIPTION / VALUE
Key.Attrib	Not used.
Payload	**Required**. MarketByPrice is represented by a **`Map`**, where each entry contains a **`FieldList`** containing information about a price point.

Table 28: Market By Price Update Message (Continued)

31.5 Usage: Market by Price Status Message

A Market By Price status message is encoded using **StatusMsg** and sent by Open Message Model interactive provider and non-interactive provider applications. This message conveys state change information associated with an item stream.

NOTE: The provider should send **Key.Name** and **ServiceName** only in the first Refresh response message, unless **MsgKeyInUpdates** is set to **true**, in which case **Key.Name** and **ServiceName** must be provided for every Status response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByPrice 8**
State	Optional. Specifies current state information associated with the data and stream.
Indications	Optional. **StatusMsgFlags.Clear_Cache**: **true**, Indicates to clear the cache.
Qos	Optional. Specifies the QoS at which the stream is provided.
PermData	Optional. Specifies new permissioning information associated with all contents on the stream.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the service that provides the data.
	NOTE: The provider application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true** on the request. Specifies the ID of the service that provides the item.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true** on the request. **Key.NameType** should match the **Key.NameType** specified in the item's request message. If **Key.NameType** is not specified, it uses the default **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Not used.

Table 29: Market By Price Status Message

32 Domain Model Usage: Market by Order Domain

32.1 Market by Order Domain Overview

The ***Market By Order*** domain provides access to Level II full order books. The list of orders is sent in the form of a **`Map`**. Each **`MapEntry`** represents one order (using the order's Id as its key) and contains a **`FieldList`** describing information related to that order (such as price, whether it is a bid/ask order, size, quote time, and market maker identifier).

NOTE: **`GenericMsg`**(s) are not supported for **MarketByOrder** Domain Models.

Refer to the following topics for details on Market by Order domain message types:

- Usage: Market by Order Request Message
- Usage: Market by Order Refresh Message
- Usage: Market by Order Update Message
- Usage: Market by Order Status Message

32.2 Market by Order Domain Examples

The following message samples illustrate the use of the Market by Order Domain.

32.2.1 Market by Order Request Message Sent

```
{
    "ID": 2,
    "Domain": "MarketByOrder",
    "Key": {
        "Name": "TRI.TO"
    }
}
```

32.2.2 Market by Order Refresh Message Received

```
[
    {
        "ID": 2,
        "Type": "Refresh",
        "Domain": "MarketByOrder",
        "Key": {
            "Service": "ELEKTRON_DD",
            "Name": "TRI.TO"
        },
        "State": {
            "Stream": "Open",
            "Data": "Ok",
            "Text": "All is well"
        },
```

```
        "Complete": false,
        "Qos": {
            "Timeliness": "Realtime",
            "Rate": "TimeConflated",
            "RateInfo": 1000
        },
        "PartNumber": 0,
        "PermData": "AwO9Z3fA",
        "SeqNumber": 11024,
        "Map": {
            "KeyType": "Buffer",
            "Summary": {
                "Fields": {
                    "PROD_PERM": 6777,
                    "DSPLY_NAME": "THOMSON REUTERS",
                    "CURRENCY": "CAD",
                    "ACTIV_DATE": "2018-04-06",
                    "PRC_QL2": " AU",
                    "LOT_SIZE_A": 100,
                    "RECORDTYPE": 113,
                    "PREF_DISP": null,
                    "RDN_EXCHD2": "TOR",
                    "LIST_MKT": "TOR",
                    "PROV_SYMB": "TRI",
                    "PR_RNK_RUL": "NOR",
                    "MNEMONIC": "TRI",
                    "MKT_STATUS": "S",
                    "TIMACT_MS": 66701289,
                    "CONTEXT_ID": 2172,
                    "DDS_DSO_ID": 8244,
                    "SPS_SP_RIC": ".[SPSTL2TRL2",
                    "BOOK_STATE": "N",
                    "HALT_REASN": null,
                    "ORD_ENT_ST": "E",
                    "MKT_OR_RUL": " ",
                    "TRD_STATUS": "N ",
                    "HALT_RSN": null
                }
            },
                    "OR_RNK_RUL": "PTS ",
            "CountHint": 333,
            "Entries": [
                {
                    "Action": "Add",
                    "Key": "MTgwNDA2MDAwMDAxNDQ1MDM5Qg==",
                    "Fields": {
                        "SEQNUM": 3335696,
                        "ORDER_ID": "000001445",
                        "ORDER_PRC": 50.47,
                        "ORDER_SIDE": "BID",
```

```
                        "ORDER_SIZE": 300,
                        "MMID": "39",
                        "PR_TIM_MS": 66600233,
                        "PR_TIM_MSP": 751,
                        "PR_DATE": "2018-04-06"
                    }
                },
                ...
                {
                    "Action": "Add",
                    "Key": "MTgwNDA2MDAwMDA0ODM2MTAxQg==",
                    "Fields": {
                        "SEQNUM": 3309457,
                        "ORDER_ID": "000004836",
                        "ORDER_PRC": 50.42,
                        "ORDER_SIDE": "BID",
                        "ORDER_SIZE": 100,
                        "MMID": "101",
                        "PR_TIM_MS": 66471013,
                        "PR_TIM_MSP": 717,
                        "PR_DATE": "2018-04-06"
                    }
                }
            ]
        }
    }
]
```

32.2.3 Market by Order Update Message Received


```
[
    {
        "ID": 2,
        "Type": "Update",
        "Domain": "MarketByOrder",
        "UpdateType": "Unspecified",
        "Key": {
            "Service": "ELEKTRON_DD",
            "Name": "TRI.TO"
        },
        "SeqNumber": 11040,
        "Map": {
            "KeyType": "Buffer",
            "Summary": {
                "Fields": {
                    "TIMACT_MS": 66702339
                }
            },
            "Entries": [
                {
                   "Action": "Add",
                    "Key": "MTgwNDA2MDAwMDEyNzUzMDAxQQ==",
                    "Fields": {
                        "ORDER_ID": "000012753",
                        "ORDER_PRC": 50.55,
                        "ORDER_SIDE": "ASK",
                        "ORDER_SIZE": 100,
                        "MMID": "1",
                        "PR_TIM_MS": 66702339,
                        "PR_TIM_MSP": 417,
                        "PR_DATE": "2018-04-06",
                        "SEQNUM": 3354677
                    }
                }
            ]
        }
    }
]
```

32.3 Usage: Market by Order Request Message

A Market By Order request message is encoded using **RequestMsg** and sent by Open Message Model consumer applications. The request specifies the name of the item in which a consumer is interested.

To receive updates, the consumer makes a "streaming" request by setting the **Request.Streaming** to **true**. If the flag is not set, the consumer is requesting a "snapshot," and the refresh should end the request. Updates may be received in either case if the refresh has multiple parts.

To stop updates, a consumer can pause an item if the provider supports this functionality. For additional details, refer to the *Enterprise Transport API Java Edition Developers Guide*.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByOrder 7**
Interactions	**Required**. • **InitialImage**: **true**, Indicates that an initial image is required. • **InterestAfterRefresh**: **true**, Indicates that a streaming request is required. • **Pause**: **true**, Indicates that a pause is required.
Indications	Optional. **ConflatedInUpdates**: **true**, indicates conflated updates is required.
Qos	Optional. Indicates the QoS at which the consumer wants the stream serviced. If both **Qos** and **WorstQos** are specified, this request can be satisfied by a range of qualities of service.
WorstQos	Optional. Used with the **Qos** member to define a range of acceptable Qualities of Service. When encountering such a range, the provider should attempt to provide the best QoS it can within that range. This should only be used on services that claim to support it via the **SupportsQosRange** item in the Source Directory response (refer to Section 4.3.1.1).
	NOTE: Enterprise Message API provides the **Request.Qos()** method to set both **Qos** and **WorstQos** depending upon the timeliness and rate values.
Priority	Optional. Indicates the class and count associated with stream priority.
ExtHdr	Not used.
Key.Service	**Required**. This should be the ID associated with the service from which the consumer wants to request the item.
	NOTE: A consumer should set either the **ServiceName** or **Key.Service** of the service, but not both.
ServiceName	**Required**. This should be the name of the service from which the consumer wishes to request data.
	NOTE: A consumer should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. When consuming from LSEG sources, **Key.NameType** is typically set to **Ric** (the "Instrument Code"). If absent, the Websocket API for Pricing Streaming and Real-Time Services assumes a setting of **Ric**.
Key.Name	**Required**. Specifies the requested item's name.
	NOTE: Not used for Batch Item requests.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.

Table 30: Market By Order Request Message

COMPONENT	DESCRIPTION / VALUE
Payload	Optional. When features such as View or Batch are leveraged, the payload can contain information relevant to that feature. For more detailed information, refer to Appendix A

Table 30: Market By Order Request Message (Continued)

32.4 Usage: Market by Order Refresh Message

A Market By Order refresh message is encoded using **RefreshMsg** and sent by Open Message Model interactive provider and non-interactive provider applications. A Market By Order refresh may be sent in multiple parts. It is possible for update and status messages to be delivered between parts of a refresh message, regardless of whether the request is streaming or non-streaming.

NOTE: The provider should send the **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true** in the Enterprise Message API configuration, then the **Key.Name** and **ServiceName** must be provided for every Refresh response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByOrder 7**
State	**Required**. The state of the stream and data.
Solicited	**Required**. Indicated whether the refresh was solicited. Available values are: • **true**: The refresh was solicited. • **false**: The refresh was unsolicited.
Indications	Optional. • **DoNotCache**: **true**, indicate do not cache this refresh message • **StatusMsgFlags.Clear_Cache**: **true**, indicate clear cache • **Complete**: **true**, indicate refresh complete
PartNum	Optional. Specifies the part number of a multi-part refresh.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	Optional.Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies permission information associated with content on this stream.
ExtHdr	Not used.
Key.Service	**Required**. Specifies the ID of the service that provides the item. **NOTE:** The provider should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. **NameType** should match the **NameType** specified in the request. If absent, **Key.NameType** is assumed to be **Ric**.
Key.Name	**Required**. **name** should match the requested item's name.
ServiceName	**Required**. Specifies the name of the service that provides the item. **NOTE:** The provider should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	**Required**. An order book is represented by a **Map**, where each entry (**MapEntry**) contains information (**FieldList**) that corresponds to an order.

Table 31: Market By Order Refresh Message

32.5 Usage: Market by Order Update Message

A Market By Order update message is encoded using **UpdateMsg** and sent by Open Message Model interactive provider and non-interactive provider applications. The provider can send an update message to add, update, or remove order information. Updates may be received between the first Refresh and the RefreshComplete. It is the consuming application's responsibility to determine if the update is applicable to the data that has previously been sent in a refresh.

NOTE: The provider should only send the **Key.Name** and **ServiceName** in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **Key.Name** and **ServiceName** must be provided for every Update response messages.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByOrder 7**
UpdateType	**Required**. Indicates the general content of the update. Typically sent as one of the following: • **Unspecified 0** • **Quote 1**
Indications	Optional: • **DoNotCache**: **true**, Indicates that the application should not cache this update message. • **DoNotConflate**: **true**, Indicates that the application should not conflate this update message.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
ConflationInfo.Count	Optional. If a provider sends a conflated update, **ConflationInfo.Count** informs the consumer as to how many updates were included in the conflation. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** to **true** in the request.
ConflationInfo.Time	Optional. If a provider sends a conflated update, **ConflationInfo.Time** informs the consumer as to the interval (in milliseconds) over which data was conflated. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** to **true** in the request.
PermData	Optional. **PermData** contains permissioning information associated only with the contents of this update.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true**. **ServiceName** specifies the name of the service that provides the data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true**. **Key.Service** specifies the ID of the service that provides the data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true**. **Key.NameType** must match the name type in the item's request message (typically **Ric**).
Key.Name	Optional (Required if **KeyInUpdates** was set to **true**). **Key.Name** specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.

Table 32: Market By Order Update Message

COMPONENT	DESCRIPTION / VALUE
Payload	**Required**. The order book is represented by a **`Map`**, where each map entry (**`MapEntry`**) holds information (**`FieldList`**) corresponding to an order.

Table 32: Market By Order Update Message (Continued)

32.6 Usage: Market by Order Status Message

A Market By Order status message is sent by Open Message Model interactive provider and non-interactive provider applications. This message conveys state change information associated with an item stream.

NOTE: The provider should only send the **Key.Name** and **ServiceName** in the first Refresh response message. However **MsgKeyInUpdates** is set to **true**, then the **Key.Name** and **ServiceName** must be provided for every Status response messages.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketByOrder 7**
State	Optional. Specifies the current state information associated with the data and stream.
Indications	Optional: **StatusMsgFlags.Clear_Cache**: **true**, Indicates to clear the cache.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	Optional. The provider may use this to change the item's **GroupId** (for details, refer to Section 4.3.1.3).
PermData	Optional. **PermData** specifies any new permissioning information associated with all of the stream's contents.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **MsgKeyInUpdates** was set to **true**). Specifies the name of the service providing data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true**. **Key.Service** specifies the ID of the service that provides the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true**. **Key.NameType** must match the name type in the item's request message. If not specified, **Key.NameType** defaults to **Ric**.
Key.Name	Optional (Required if **KeyInUpdates** was set to **true**). **Key.Name** specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Not used.

Table 33: Market By Order Status Message

33 Domain Model Usage: Market Maker Domain

33.1 Market Maker Domain Overview

The ***Market Maker*** domain provides access to market maker quotes and trade information. The list of market makers is sent in the form of a **`Map`**. Each **`MapEntry`** represents one market maker (using that market maker's ID as its key) and contains a **`FieldList`** describing information such as that market maker's bid and ask prices, quote time, and market source.

NOTE: **`GenericMsg`**(s) are not supported for the **MarketMaker** Domain Model.

Refer to the following topics for details on Market Maker domain message types:

- Usage: Market Maker Request Message
- Usage: Market Maker Refresh Message
- Usage: Market Maker Update Message
- Usage: Market Maker Status Message

33.2 Market Maker Domain Examples

The following message samples illustrate the use of the Market Maker Domain.

33.2.1 Market Maker Request Message Sent

```
{
    "ID": 2,
    "Domain": "MarketMaker",
    "Key": {
        "Name": "MSFT.O"
    }
}
```

33.2.2 Market Maker Refresh Message Received

```
{
    "ID": 2,
    "Type": "Refresh",
    "Domain": "MarketMaker",
    "Key": {
        "Service": "ELEKTRON_DD",
        "Name": "MSFT.O"
    },
    "State": {
        "Stream": "Open",
        "Data": "Ok",
        "Text": "All is well"
    },
    "Complete": false,
```

```
    "Qos": {
        "Timeliness": "Realtime",
        "Rate": "TimeConflated",
        "RateInfo": 1000
    },
    "PermData": "AwO9MFbA",
    "SeqNumber": 3552,
    "Map": {
        "KeyType": "Buffer",
        "Summary": {
            "Fields": {
                "PROD_PERM": 3056,
                "DSPLY_NAME": "MICROSOFT CP",

                ...
                     (Additional Entries)
                ...

                "IPO_QR_CD": null,
                "IPO_QR_MS": null
            }
        },
        "CountHint": 58,
        "Entries": [
            {
                "Action": "Add",
                "Key": "Q1RETA==",
                "Fields": {
                    "BID": 90.53,
                    "ASK": 101.1,
                    "BIDSIZE": 100,
                    "ASKSIZE": 100,
                    "MKT_MKR_NM": "CITADEL DERIVAT",
                    "MMID": "CTDL",
                    "ASK_TIM_MS": 49739010,
                    "TIMACT_MS": 49739010,
                    "BID_TIM_MS": 48604536,
                    "PRIMARY_MM": "Y",
                    "MM_MODE": "N ",
                    "MM_STATE": "A ",
                    "PR_TIM_MS": 25623489,
                    "PR_DATE": "2018-05-07"
                }
            },

            ...
                     (Additional Entries)
            ...

            {
```

```
                "Action": "Add",
                "Key": "U1RGTA==",
                "Fields": {
                    "BID": 88.3,
                    "ASK": 103.65,
                    "BIDSIZE": 100,
                    "ASKSIZE": 100,
                    "MKT_MKR_NM": "STIFEL NICOLAUS",
                    "MMID": "STFL",
                    "ASK_TIM_MS": 49501183,
                    "TIMACT_MS": 49501185,
                    "BID_TIM_MS": 49501185,
                    "PRIMARY_MM": "Y",
                    "MM_MODE": "N ",
                    "MM_STATE": "A ",
                    "PR_TIM_MS": 25624734,
                    "PR_DATE": "2018-05-07"
                }
            },
            {
                "Action": "Add",
                "Key": "RUdNVA==",
                "Fields": {
                    "BID": 88.3,
                    "ASK": 103.65,
                    "BIDSIZE": 100,
                    "ASKSIZE": 100,
                    "MKT_MKR_NM": "EGMT",
                    "MMID": "EGMT",
                    "ASK_TIM_MS": 49501182,
                    "TIMACT_MS": 49501184,
                    "BID_TIM_MS": 49501184,
                    "PRIMARY_MM": "Y",
                    "MM_MODE": "N ",
                    "MM_STATE": "A ",
                    "PR_TIM_MS": 25623635,
                    "PR_DATE": "2018-05-07"
                }
            }
        ]
    }
}
```

33.2.3 Market Maker Update Message Received


```
{
    "ID": 2,
    "Type": "Update",
    "Domain": "MarketMaker",
    "UpdateType": "Unspecified",
    "Key": {
        "Service": "ELEKTRON_DD",
        "Name": "MSFT.O"
    },
    "SeqNumber": 3568,
    "Map": {
        "KeyType": "Buffer",
        "Entries": [
            {
                "Action": "Update",
                "Key": "TlNEUQ==",
                "Fields": {
                    "BID": 96.42,
                    "BIDSIZE": 405,
                    "BID_TIM_MS": 56143000,
                    "TIMACT_MS": 56143000,
                    "MMID": "NSDQ"
                }
            }
        ]
    }
}
```

33.3 Usage: Market Maker Request Message

A Market Maker request message is encoded using **RequestMsg** and sent by Open Message Model consumer applications. The request specifies the name of an item in which the consumer is interested.

To receive updates, a consumer can make a "streaming" request by setting the **Request.Streaming** to **true**. If the flag is not set, the consumer requests a "snapshot," and the final part of the refresh indicates all responses have been received for the snapshot. Updates may be received in either case if the refresh has multiple parts.

To stop updates, a consumer can pause an item (if the provider supports this functionality). For more information, refer to the *Enterprise Transport API Java Edition Developers Guide*.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketMaker 9**
Interactions	**Conditional**. Use the appropriate interactions according to your messaging needs: • **InitialImage**: **true**, indicates that an initial image is required. • **InterestAfterRefresh**: **true**, indicates that a streaming request is required. • **Pause**: **true**, indicates that a pause is required.
Indications	Optional. **ConflatedInUpdates**: **true**, indicates that conflated updates are required. Batch and View request are specified in the **Payload**.
Qos	Optional. Indicates the QoS at which the consumer wants the stream serviced. If both **Qos** and **WorstQos** are specified, this request can be satisfied by a range of QoS.
WorstQos	Optional. Used with **Qos** to define a range of acceptable QoS. If the provider encounters such a range, it should attempt to provide the best possible QoS within that range. This should only be used on services that claim to support it via the **SupportsQosRange** item in the Source Directory response (for details, refer to Section 4.3.1.1).
	NOTE: Enterprise Message API provides the **Request.Qos()** method to set both **Qos** and **WorstQos** depending upon the timeliness and rate values.
Priority	Optional. Indicates the class and count associated with stream priority.
ExtHdr	Not used.
Key.Service	**Required**. Specifies the ID of the service that provides the requested item.
	NOTE: The consumer application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. When consuming from LSEG sources, **Key.NameType** is typically set to **Ric** (the "Instrument Code"). If absent, its value reverts to the default, which is **Ric**.
Key.Name	**Required**. Specifies the name of the requested item.
	NOTE: Not used for Batch Item request.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Optional. When features such as View or Batch are leveraged, the payload can contain information relevant to that feature. For more details, refer to Appendix A.

Table 34: Market Maker Request Message

33.4 Usage: Market Maker Refresh Message

A Market Maker refresh message is encoded using **RefreshMsg** and sent by Open Message Model interactive provider and non-interactive provider applications.

The Market Maker refresh can be sent in multiple parts. Keep in mind that both update and status messages can be delivered between parts of a refresh message, regardless of streaming or non-streaming request.

NOTE: The provider should send the **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true** in the Enterprise Message API configuration, then the **Key.Name** and **ServiceName** must be provided for every Refresh response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketMaker**
State	**Required**. Indicates the state of the stream and data.
PartNum	Optional. Specifies the part number of a multi-part refresh.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	**Required**. Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies permission information associated with this stream's content.
ExtHdr	Not used.
Key.Service	**Required**. Specifies the ID of the service that provides the item.
Key.NameType	Optional. **NameType** should match the **NameType** specified in the request. If absent, **Key.NameType** defaults to **Ric**.
Key.Name	**Required**. A symbol for the Market Maker item
Key.Filter	Not used.
Key.Identifier	Not used.
Payload	**Required**. is represented by a **Map**, where each entry contains a **FieldList** which has information about a market maker.

Table 35: Market Maker Refresh Message

33.5 Usage: Market Maker Update Message

A Market Maker update message is encoded using **`UpdateMsg`** and sent by Open Message Model interactive provider and non-interactive provider applications. Updates will not be received before images, and a true snapshot is supported.

The provider can send an update message to add, update, or remove market maker information.

NOTE: The provider should send the **`Key.Name`** and **`ServiceName`** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **`Key.Name`** and **`ServiceName`** must be provided for every Update response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketMaker 9**
UpdateType	**Required**. Indicates the general content of the update. Typically sent as one of the following: • **Unspecified 0** • **Quote 1**
Indications	Optional: • **`DoNotCache`**: **true**, specifies that the update message should not be cached. • **`DoNotConflate`**: **true**, specifies that the update message should not be conflated.
PartNum	Not used.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
ConflationInfo.Count	Optional. If a provider sends a conflated update, **`ConflationInfo.Count`** specifies how many updates are in the conflation. The consumer indicates interest in this information by setting **ConfInfoInUpdates** to true in the request.
ConflationInfo.Time	Optional. If a provider sends a conflated update, **`ConflationInfo.Time`** specifies the time interval (in milliseconds) over which data is conflated. The consumer indicates interest in this information by setting **ConfInfoInUpdates** to true in the request.
PermData	Optional. Specifies permissioning information associated only with the contents of this update.
ExtHdr	Not used.
ServiceName	**Conditional**. **`ServiceName`** is required if **KeyInUpdates** was set to **true**. **`ServiceName`** specifies the name of the service that provides the data. **NOTE:** The provider application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.Service	**Conditional**. **`Key.Service`** is required if **KeyInUpdates** was set to **true**. **`Key.Service`** specifies the ID of the service that provides the item. **NOTE:** The provider application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.NameType	**Conditional**. **`Key.NameType`** is required if **KeyInUpdates** was set to **true**. **`Key.NameType`** must match the name type in the item's request message (typically **Ric**). If absent, **`Key.NameType`** defaults to **Ric**.
Key.Name	**Conditional**. **`Key.Name`** is required if **KeyInUpdates** was set to **true**. **`Key.Name`** specifies the name of the item being provided.
Key.Filter	Not used.

Table 36: Market Maker Update Message

COMPONENT	DESCRIPTION / VALUE
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	**Required**. A Market Makeris represented by a **Map**, where each entry (**MapEntry**) contains a **FieldList** which in turn contains information about a market maker.

Table 36: Market Maker Update Message (Continued)

33.6 Usage: Market Maker Status Message

A Market Maker status message is encoded and sent by Open Message Model interactive provider and non-interactive provider applications. This message conveys state change information associated with an item stream.

NOTE: The provider should send the **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **Key.Name** and **ServiceName** must be provided for every Status response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **MarketMaker 9**
State	Optional. Specifies current state information associated with the data and stream.
Indications	Optional. **StatusMsgFlags.Clear_Cache**: **true**, indicates that the application should clear the cache.
Qos	Optional. Specifies the QoS at which the stream is provided.
PermData	Optional. Specifies new permissioning information associated with all of the stream's contents.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true**. **ServiceName** specifies the name of the service that provides the data. **NOTE:** The provider application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true**. **Key.Service** specifies the ID of the service that provides the item.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true**. **Key.NameType** must match the name type in the item's request message (typically **Ric**). If absent, **Key.NameType** defaults to **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true** **Key.Name** specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Not used.

Table 37: Market Maker Status Message

34 Domain Model Usage: Yield Curve Domain

34.1 Yield Curve Domain Overview

The ***Yield Curve*** domain shows the relation between the interest rate and the term (time to maturity) associated with the debt of a borrower. The shape of a yield curve can help give an idea of future economic activity and interest rates. Information is sent as a **`FieldList`**, where some **`FieldEntry`**'s can contain more complex types such as **`Vector`**, **`Array`**, or **`ElementList`**.

This chapter documents the Yield Curve domain as provided by the LSEG Real-Time Advanced Transformation Server.

NOTE: The **YieldCurve** Domain Model does not support **`GenericMsg`**(s).

Refer to the following topics for details on Yield Curve domain message types:

- Usage: Yield Curve Request Message
- Usage: Yield Curve Refresh Message
- Usage: Yield Curve Update Message
- Usage: Yield Curve Status Message

34.2 Yield Curve Domain Examples

The following message samples illustrate the use of the Yield Curve Domain.

34.2.1 Yield Curve Request Message Sent

```
{
    "ID": 2,
    "Domain": "YieldCurve",
    "Key": {
        "Service": "ATS201_1",
        "Name": "BASIC"
    }
}
```

34.2.2 Yield Curve Refresh Message Received

```
{
    "ID": 2,
    "Type": "Refresh",
    "Domain": "YieldCurve",
    "Key": {
        "Service": "ATS201_1",
        "Name": "BASIC"
    },
    "State": {
        "Stream": "Open",
        "Data": "Ok",
        "Text": "Item Refresh Completed\u0000"
```

```
    },
    "Qos": {
        "Timeliness": "Realtime",
        "Rate": "TickByTick",
        "Dynamic": true
    },
    "ClearCache": false,
    "PermData": "AwEA0AybITw=",
    "Fields": {
        "CASH_BASIS": "ACT/360",
        "CS_INT_MTH": "Linear",
        "TRADE_DATE": "2018-05-07",
        "TIMACT": "20:34:47.315",
        "CRV_ID": 1,
        "CRV_NAME": "BASIC",
        "CCY_CODE": null,
        "CRV_TYPE": "Swap",
        "CRV_STYPE": "Standard",
        "CRV_DATE": "2018-05-08",
        "CITIES": "JP",
        "VAL_DATE": "2018-05-08",
        "SETTL_DATE": "2018-05-08",
        "CRV_ALGTHM": "LSEG Real-Time Advanced Transformation System",
        "INTER_MTHD": null,
        "EXTRP_MTHD": null,
        "CC_METHOD": "Bootstrap",
        "ROLL_CONV": "Modified Following",
        "ZC_BASIS": "ACT/360",
        "SPOT_LAG": 0,
        "DSCT_FACT": "Compound",
        "DSCT_BASIS": "ACT/360",
        "CURVE_STS": "Created",
        "USER_ID": "ADMIN",
        "MOD_USERID": "ADMIN",
        "CRT_DATE": "2015-05-16",
        "MOD_DATE": "2015-05-16",
        "COMMENT": null,
        "FWD_BASIS": "ACT/360",
        "FT_INT_MTH": "Linear",
        "CASH_RATES": {
            "Summary": {
                "Fields": {
                    "TENORS": {
                        "Type": "AsciiString",
                        "Data": [
                            "1M",
                            "3M"
                        ]
                    }
                }
```

```
            },
            "Entries": [
                {
                    "Index": 0,
                    "Action": "Set",
                    "Fields": {
                        "CASH_SDATE": "2018-05-08",
                        "CASH_MDATE": "2018-06-08",
                        "CASH_RATE": 109.08,
                        "CASH_SRC": "JPY="
                    }
                },
                {
                    "Index": 1,
                    "Action": "Set",
                    "Fields": {
                        "CASH_SDATE": "2018-05-08",
                        "CASH_MDATE": "2018-08-08",
                        "CASH_RATE": 109.07,
                        "CASH_SRC": "JPY="
                    }
                }
            ]
        },
        "YLD_CURVE": {
            "Summary": {
                "Fields": {
                    "TENORS": {
                        "Type": "AsciiString",
                        "Data": [
                            "1M",
                            "3M"
                        ]
                    }
                }
            },
            "Entries": [
                {
                    "Index": 0,
                    "Action": "Set",
                    "Fields": {
                        "YCT_DATE": "2018-06-08",
                        "YCT_ZRATE": 183.64926925186,
                        "YCT_DISFAC": 0.91413527373781
                    }
                },
                {
                    "Index": 1,
                    "Action": "Set",
                    "Fields": {
```

```
                        "YCT_DATE": "2018-08-08",
                        "YCT_ZRATE": 161.71957450513,
                        "YCT_DISFAC": 0.78202319828372
                    }
                }
            ]
        }
    }
}
```

34.2.3 Yield Curve Update Message Received

```
{
    "ID": 2,
    "Type": "Update",
    "Domain": "YieldCurve",
    "UpdateType": "Unspecified",
    "Key": {
        "Service": "ATS201_1",
        "Name": "BASIC"
    },
    "Fields": {
        "CRV_ID": 4,
        "TRADE_DATE": "2018-05-07",
        "TIMACT": "20:34:54.317",
        "CASH_RATES": {
            "Entries": [
                {
                    "Index": 0,
                    "Action": "Update",
                    "Fields": {
                        "CASH_RATE": 109.08
                    }
                },
                {
                    "Index": 1,
                    "Action": "Update",
                    "Fields": {
                        "CASH_RATE": 109.06
                    }
                }
            ]
        },
        "YLD_CURVE": {
            "Entries": [
                {
                    "Index": 0,
                    "Action": "Update",
```

```
                    "Fields": {
                        "YCT_DATE": "2018-06-08",
                        "YCT_ZRATE": 183.64926925186,
                        "YCT_DISFAC": 0.91413527373781
                    }
                },
                {
                    "Index": 1,
                    "Action": "Update",
                    "Fields": {
                        "YCT_DATE": "2018-08-08",
                        "YCT_ZRATE": 161.69910802302,
                        "YCT_DISFAC": 0.78203882735885
                    }
                }
            ]
        }
    }
}
```

34.3 Usage: Yield Curve Request Message

A Yield Curve request message is encoded using **`RequestMsg`** and sent by Open Message Model consumer applications. The request specifies the name and attributes of the curve in which the consumer is interested.

To receive updates, the consumer makes a “streaming” request by setting the **Request.Streaming** to **true**. If the flag is not set, the consumer requests a “snapshot,” and the final part of the refresh (i.e., the refresh has the **Complete** flag set) indicates all responses have been received for the snapshot. Updates may be received in either case if the refresh has multiple parts.

To stop updates, a consumer can pause an item (if the provider supports the pause feature). For additional details, refer to the *Enterprise Transport API Java Edition Developers Guide*.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **YieldCurve 22**
Interactions	**Conditional**. • **`InitialImage`**: **true**, requests an initial image. • **`InterestAfterRefresh`**: **true**, requests streaming updates. • **`Pause`**: **true**, requests that the application pause the item.
Indications	Optional. **`ConflatedInUpdates`**: **true**, requests that the application send conflated updates. Batch and View request are specified in the **`Payload`**.
Qos	Optional. Indicates the QoS at which the consumer wants the stream serviced. If both **`Qos`** and **`WorstQos`** are specified, this request can be satisfied by a range of QoS.

Table 38: Yield Curve Request Message

COMPONENT	DESCRIPTION / VALUE
WorstQos	Optional. Used with the **Qos** member to define a range of acceptable QoS. When the provider encounters such a range, it should attempt to provide the best QoS it can within that range. **WorstQos** should only be used on services that claim to support it via the **SupportsQosRange** item in the Source Directory response (refer to Section 4.3.1.1).
	NOTE: Enterprise Message API provides the **Request.Qos()** method to set both **Qos** and **WorstQos** depending upon the timeliness and rate values.
Priority	Optional. Indicates class and count associated with stream priority.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service from which the consumer wishes to request the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Required**. Specifies the ID of the service that provides the requested item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. When consuming from LSEG sources, typically set to **Ric** (the “Instrument Code”). If this is not specified, **Key.NameType** defaults to **Ric**.
Key.Name	**Required**.Specifies the name of the requested item.
	NOTE: Not used for Batch Item request.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Optional. When leveraging such features as View or Batch, the payload can contain information relevant to that feature. For more information, refer to Appendix A.

Table 38: Yield Curve Request Message (Continued)

34.4 Usage: Yield Curve Refresh Message

A Yield Curve Refresh Message is encoded using **RefreshMsg** and sent by Open Message Model provider and non-interactive provider applications. This message sends all currently available information about the item to the consumer.

FieldList in the payload should include all fields that might be present in subsequent updates, even if those fields are currently blank. When responding to a View request, this refresh should contain all fields requested by the specified view. If for any reason the provider wishes to send new fields, it must first send an unsolicited refresh with both the new and currently-present fields.

NOTE: The provider should send the **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true** in the Enterprise Message API configuration, then the **Key.Name** and **ServiceName** must be provided for every Refresh response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **YieldCurve 22**

Table 39: Yield Curve Refresh Message

COMPONENT	DESCRIPTION / VALUE
State	**Required**. Includes the state of the stream and data.
Solicited	**Required**. Indicates whether the refresh was solicited. Available values are: • **true**: The message was solicited. • **false**: The message was unsolicited.
Indications	**Conditional**. • **`DoNotCache`**: **true**, indicates that the application should not cache this refresh message. • **`StatusMsgFlags.Clear_Cache`**: **true**, indicates that the application should clear the cache. • **`Complete`**: **true**, indicates that the message is the final one in the refresh.
PartNum	Optional. Specifies the part number of a multi-part refresh.
Qos	Optional. Specifies the QoS at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	**Required**. Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies permission information associated with content on this stream.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service that provides the item.
	NOTE: The application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.Service	**Required**. Specifies the ID of the service that provides the item.
	NOTE: The application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.NameType	Optional. Should match the **`NameType`** specified in the request. If this is not specified, **`NameType`** defaults to **Ric**.
Key.Name	**Required**. This should match the requested name.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	**Required**. This should consist of a **`FieldList`** containing all fields associated with the item. Some **`FieldEntry`**s are sent as more complex types such as **`Vector`** and **`DataTypes.Array`**. Encoding and decoding applications should be aware of this and ensure proper handling of these types.

Table 39: Yield Curve Refresh Message (Continued)

34.5 Usage: Yield Curve Update Message

A Yield Curve Update Message is encoded using **`UpdateMsg`** and sent by Open Message Model provider and non-interactive provider applications. It conveys any changes to an item's data. Updates may be received between the first Refresh and the RefreshComplete. It is the consuming application's responsibility to determine if the update is applicable to the data that has previously been sent in a refresh.

NOTE: The provider should send the **`Key.Name`** and **`ServiceName`** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **`Key.Name`** and **`ServiceName`** must be provided for every Refresh response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **YieldCurve 22**
UpdateType	**Required**. Indicates the general content of the update. Typically sent as one of the following: • **Unspecified 0** • **Quote 1**
Indications	**Conditional**. • **`DoNotCache`**: **true**, indicates that the application should not cache this update message. • **`DoNotConflate`**: **true**, indicates that the application should not conflate the update message.
SeqNumber	Optional. A user-specified, item-level sequence number which the application can use to sequence messages in this stream.
PartNum	Not used.
ConflationInfo.Count	Optional. If the provider sends a conflated update, **`ConflationInfo.Count`** specifies how many updates are in the conflation. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** to **true** in the request.
ConflationInfo.Time	Optional. If a provider is sending a conflated update, **`ConflationInfo.Time`** specifies the time interval (in milliseconds) over which data is conflated. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** to **true** in the request.
PermData	Optional. Permissioning information associated with only the contents of this update.
ExtHdr	Not used.
ServiceName	**Conditional**. **`ServiceName`** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the service that provides the data.
	NOTE: The application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.Service	**Conditional**. **`Key.Service`** is required if **KeyInUpdates** was set to **true** on the request. Specifies the ID of the service that provides the item.
	NOTE: The application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.NameType	**Conditional**. **`Key.NameType`** is required if **KeyInUpdates** was set to **true** on the request. Should match the **`Key.NameType`** specified on the request. If this is not specified, **`Key.NameType`** defaults to **Ric**.
Key.Name	**Conditional**. **`Key.Name`** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	**Required**. This should consist of a **`FieldList`** containing all fields associated with the item. Some **`FieldEntry`**s are sent as more complex types such as **`Vector`** and **`DataTypes.Array`**. Encoding and decoding applications should be aware of this and ensure proper handling of these types.

Table 40: Yield Curve Update Message

34.6 Usage: Yield Curve Status Message

A Yield Curve status message is encoded using **StatusMsg** and sent by Open Message Model interactive provider and non-interactive provider applications. This message conveys state change information associated with an item stream.

NOTE: The provider should send the **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **Key.Name** and **ServiceName** must be provided for every Status response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **YieldCurve 22**
State	Optional. Current state information associated with the data and stream.
Indications	Optional. **StatusMsgFlags.Clear_Cache**: **true**, indicates that the cache should be cleared.
Qos	Optional. Specifies the QoS at which the stream is provided.
PermData	Optional. Specifies new permissioning information associated with all contents on the stream.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true** on the request. Specifies the name of the service that provides the data.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true** on the request. Specifies the ID of the service that provides the item.
	NOTE: The application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true** on the request Should match the **Key.NameType** specified on the request. If this is not specified, **Key.NameType** defaults to **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true** on the request Specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Not used.

Table 41: Yield Curve Status Message

35 Domain Model Usage: Symbol List Domain

35.1 Symbol List Domain Overview

The ***Symbol List*** domain provides access to a set of symbol names, typically from an index, service, or cache. Content is encoded as a **`Map`**, with each symbol represented by a map entry and where the symbol name is the entry key. An entry's payload is optional, but when present the payload is a **`FieldList`** that contains additional cross-reference information such as permission information, name type, or other venue-specific content.

NOTE: **`GenericMsg`**(s) are not supported for **SymbolList** Domain Model.

Refer to the following topics for details on Symbol List domain message types:

- Usage: Symbol List Request Message
- Usage: Symbol List Refresh Message
- Usage: Symbol List Update Message
- Usage: Symbol List Status Message

35.2 Symbol List Domain Examples

The following message samples illustrate the use of the Symbol List Domain.

35.2.1 Symbol List Request Message Sent

```
{
    "ID": 2,
    "Domain": "SymbolList",
    "Key": {
        "Name": ".AV.N"
    }
}
```

35.2.2 Symbol List Refresh Message Received

```
{
    "ID": 2,
     "Type": "Refresh",
     "Domain": "SymbolList",
     "Key": {
         "Service": "ELEKTRON_DD",
         "Name": ".AV.N"
     },
     "State": {
         "Stream": "Open",
         "Data": "Ok",
         "Text": "All is well"
     },
```

```
    "Qos": {
        "Timeliness": "Realtime",
        "Rate": "TimeConflated",
        "RateInfo": 1000
    },
    "PermData": "AwO9YsA=",
    "SeqNumber": 33104,
    "Map": {
        "KeyType": "Buffer",
        "Summary": {
            "Fields": {
                "PROD_PERM": 62,
                "RDNDISPLAY": 173,
                "DSPLY_NAME": "TOP 25 BY VOLUME",
                "RDN_EXCHID": "NYS",
                "TIMACT": "16:57:54",
                "ACTIV_DATE": "2018-05-04",
                "NUM_MOVES": 832,
                "OFFCL_CODE": "000000000000",
                "RECORDTYPE": 117,
                "DSO_ID": null,
                "RDN_EXCHD2": "NYS",
                "TIMACT1": "16:57:54",
                "MKT_SECTOR": "0",
                "DDS_DSO_ID": 8287,
                "SPS_SP_RIC": ".[SPSNYSE1VAE1"
            }
        },
        "CountHint": 25,
        "Entries": [
            {
                "Action": "Add",
                "Key": "Ri5O",
                "Fields": {
                    "RANK_POS": 8
                }
            },
            {
                "Action": "Add",
                "Key": "V0ZULk4=",
                "Fields": {
                    "RANK_POS": 21
                }
            },

            ...
                (Additional Entries)
            ...

            {
```

```
                "Action": "Add",
                "Key": "Tk9LLk4=",
                "Fields": {
                    "RANK_POS": 7
                }
            },
            {
                "Action": "Add",
                "Key": "Uy5O",
                "Fields": {
                    "RANK_POS": 20
                }
            }
           ]
    }
}
```

35.2.3 Symbol List Update Message Received

```
{
    "ID": 2,
    "Type": "Update",
    "Domain": "SymbolList",
    "UpdateType": "MarketDigest",
    "Key": {
        "Service": "ELEKTRON_DD",
        "Name": ".AV.N"
    },
    "SeqNumber": 33136,
    "Map": {
        "KeyType": "Buffer",
        "Summary": {
            "Fields": {
                "TIMACT": "16:58:24",
                "NUM_MOVES": 834,
                "TIMACT1": "16:58:24"
            }
        }
    }
}
```

35.3 Usage: Symbol List Request Message

A Symbol List request message is encoded and sent by Open Message Model consumer applications.

The consumer can make a streaming request (set **Request.Streaming** to **true**) to receive updates, typically associated with item additions or removals from the list.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **SymbolList 10**
Interactions	**Conditional**. • **`InitialImage`**: **true**, indicates that an initial image is required. • **`InterestAfterRefresh`**: **true**, indicates that a streaming request is required. • **`Pause`**: **true**, indicates that a pause is required.
Indications	Optional. **`ConflateInUpdates`**: **true**, indicates that conflated updates are required. Batch and View requests are specified in the **`Payload`**.
Qos	Not used.
WorstQos	Not used.
Priority	Optional. Indicates class and count associated with stream priority.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service from which the consumer wants to request the item.
	NOTE: The consumer application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.Service	**Required**. Specifies the ID of the service that provides the requested item.
	NOTE: The consumer application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.NameType	Optional. **`Key.NameType`** should match name type specified in the request. When consuming from LSEG sources, **`Key.NameType`** is typically set to **Ric** (the “Instrument Code”). If absent, **`Key.NameType`** defaults to **Ric**.
Key.Name	**Required**. Specifies the name of the requested item.
	NOTE: Not used for Batch Item requests.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Optional. When leveraging such features as View, Batch, or behaviors related to the Symbol List Request, the payload can contain information relevant to that feature. For more detailed information, refer to Appendix A.

Table 42: Symbol List Request Message

35.4 Usage: Symbol List Refresh Message

A Symbol List refresh Message is encoded using **RefreshMsg** and sent by Open Message Model provider and non-interactive provider applications. This message sends a list of item names to the consumer.

A Symbol List refresh can be sent in multiple parts. Update and status messages can be delivered between parts of a refresh message, regardless of streaming or non-streaming request.

NOTE: The provider should send the **Key.Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true** in the Enterprise Message API configuration, then the **Key.Name** and **ServiceName** must be provided for every Refresh response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **SymbolList 10**
State	**Required**. Indicates the state of the stream and data.
Solicited	**Required**. Indicates whether the refresh was solicited. Available values are: • **true**: The message was solicited. • **false**: The message was unsolicited.
Indications	**Conditional**. • **DoNotCache**: **true**, requests that the application not cache this refresh message. • **StatusMsgFlags.Clear_Cache**: **true**, requests that the application clear the cache. • **Complete**: **true**, indicates that this message completes the refresh.
PartNum	Optional. Specifies the part number of a multi-part refresh.
Qos	Optional. Specifies the quality of service at which the stream is provided.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
GroupId	Optional. Associates the item with an Item Group (refer to Section 4.3.1.3).
PermData	Optional. Specifies the permission information associated with content on this stream.
ExtHdr	Not used.
ServiceName	**Required**. Specifies the name of the service from which the consumer wants to request the item.
	NOTE: The consumer application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Required**. Specifies the ID of the service that provides the item.
	NOTE: The consumer application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	Optional. **NameType** should match the **NameType** specified in the request. If absent, it is assumed to be **Ric**.
Key.Name	**Required**. **Key.Name** should match the requested name.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.

Table 43: Symbol List Refresh Message

COMPONENT	DESCRIPTION / VALUE
Payload	**Required**. The payload contains a **`Map`** where each entry represents an item in the list. Each map entry contains a **`FieldList`** or **`ElementList`** with additional info about that item.

Table 43: Symbol List Refresh Message (Continued)

35.5 Usage: Symbol List Update Message

A Symbol List Update Message is encoded using **UpdateMsg** and sent by Open Message Model provider and non-interactive provider applications. It adds or removes items from the list. Updates will not be received before images, and a true snapshot is supported.

NOTE: The provider should send the **Name** and **ServiceName** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to true, then the **Name** and **ServiceName** must be provided for every Update response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **SymbolList 10**
Indications	**Conditional**. • **DoNotCache**: **true**, indicates to not cache this update message. • **DoNotConflate**: **true**, indicates to not conflate the update message.
Qos	Optional. Specifies the quality of service at which the stream is provided.
UpdateType	Not used.
SeqNumber	Optional. A user-specified, item-level sequence number which can be used by the application for sequencing messages within this stream.
ConflationInfo.Count	Optional. If a provider sends a conflated update, **ConflationInfo.Count** specifies how many updates are in the conflation. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** is set to **true** in the request.
ConflationInfo.Time	Optional. If a provider sends a conflated update, **ConflationInfo.Time** specifies the time interval (in milliseconds) over which data is conflated. The consumer indicates interest in this information by setting the **ConfInfoInUpdates** is set to **true** in the request.
PermData	Optional. Specifies the permission information associated with only the contents of this update.
ExtHdr	Not used.
ServiceName	**Conditional**. **ServiceName** is required if **KeyInUpdates** was set to **true**. **ServiceName** specifies the name of the service that provides the data. **NOTE:** The provider application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.Service	**Conditional**. **Key.Service** is required if **KeyInUpdates** was set to **true**. Specifies the ID of the service that provides the item. **NOTE:** The provider application should set either the **ServiceName** or **Key.Service** of the service, but not both.
Key.NameType	**Conditional**. **Key.NameType** is required if **KeyInUpdates** was set to **true**. Set this to match the **Key.NameType** in the item's request message (typically **Ric**). If absent, it is assumed to be **Ric**.
Key.Name	**Conditional**. **Key.Name** is required if **KeyInUpdates** was set to **true**. Specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.

Table 44: Symbol List Update Message

COMPONENT	DESCRIPTION / VALUE
Payload	**Required**. The payload contains a **`Map`**, where each entry represents an item in the list. Each map entry contains a **`FieldList`** with additional information about that item.

Table 44: Symbol List Update Message (Continued)

35.6 Usage: Symbol List Status Message

A Symbol List status message is encoded using **`StatusMsg`** and sent by Open Message Model interactive provider and non-interactive provider applications. This message conveys state change information associated with an item stream.

NOTE: The provider should send the **`Key.Name`** and **`ServiceName`** only in the first Refresh response message. However if **MsgKeyInUpdates** is set to **true**, then the **`Key.Name`** and **`ServiceName`** must be provided for every Status response message.

COMPONENT	DESCRIPTION / VALUE
Domain	**Required**. **SymbolList 10**
State	Optional. Current state information associated with the data and stream.
Indications	**Conditional**. **`StatusMsgFlags.Clear_Cache`**: **true**, indicates to clear the cache.
Qos	Optional. Specifies the quality of service at which the stream is provided.
GroupId	Optional. The provider can use this to change the item's **`GroupId`**.
PermData	Optional. Specifies new permissioning information associated with the stream's contents.
ExtHdr	Not used.
ServiceName	**Conditional**. **`ServiceName`** is required if **KeyInUpdates** was set to **true**. **`ServiceName`** specifies the name of the service that provides the data.
	NOTE: The provider application should set either the **`ServiceName`** or **`Key.Service`** of the service, but not both.
Key.Service	**Conditional**. **`Key.Service`** is required if **KeyInUpdates** was set to **true**. Specifies the ID of the service that provides the item.
Key.NameType	**Conditional**. **`Key.NameType`** is required if **KeyInUpdates** was set to **true**. **`Key.NameType`** should match the name type specified on the request. If it is not specified, **`Key.NameType`** defaults to **Ric**.
Key.Name	**Conditional**. **`Key.Name`** is required if **KeyInUpdates** was set to **true**. Specifies the name of the item being provided.
Key.Filter	Not used.
Key.Identifier	Not used.
Key.Attrib	Not used.
Payload	Not used.

Table 45: Symbol List Status Message

LSEG DATA & ANALYTICS

Document ID: WSA100LI.240
Date of issue: November 2024