Background Information

This section provides important information about the Coinext exchange software..

Standard Response Object and Common Error Codes

A response to an API call usually consists of a specific response object (as documented in this guide), but both successful and unsuccessful responses may consist of a generic response object that verifies that the call was received; the response to an unsuccessful call provides an error code. A generic response looks like:

```
{
  "result": true,
  "errormsg": "",
  "errorcode": 0,
  "detail": "",
}
```

Where:

String	Value
result	**Boolean.** If the call has been successfully received by the Order Management System, result is *true*; otherwise, it is *false*.

Table continued from page 2

errormsg	**string.** A successful receipt of the call returns null; the *errormsg* parameter for an unsuccessful call returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the call returns 0. An unsuccessful receipt of the call returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. The content of this parameter is usually *null*.

Products and Instruments

In Coinext software, a *product* is an asset that is tradable or paid out. A product might be a currency or a commodity or something else. For example, a product might be a US Dollar or a New Zealand Dollar or a BitCoin or an ounce of gold. Fees are denominated in products. (Products may also be referred to as *assets* in the API calls.)

An instrument is a pair of exchanged products (or fractions of them). For example, US Dollars and an ounce of gold, or an ounce of gold and BitCoins. In conventional parlance, a stock or a bond is called an instrument, but implicit in that is the potential exchange of one product for another (stock for dollars). Coinext software thinks of that exchange as explicit.

Quotes and Orders

The Coinext API includes calls related to both quotes and orders.

- A quote expresses a willingness to buy or sell at a given price.
- An order is a directive to buy or sell.

In Version 2.23.9 or earlier of the Coinext matching engine software, quotes and orders are synonymous. They both can result in a sell or a buy. This is because the matching engine (like most matching engines) requires a "firm quote" — a guaranteed bid or ask. For both quotes and orders, trading priority is the same, and no preference is given one over the other. In code, the matching engine flags a quote for eventual regulatory and compliance rules, but as far as current software operation and trade execution, they behave equivalently.

Quoting is not enabled for the retail end user of the Coinext software. Only registered market participants or market makers may quote.

Your trading venue may offer quotes separately from orders.

Best practices: Use the order-related API calls in preference to quote-related calls unless you specifically require the quote-related calls.

Order-related API calls	Quote-related API calls
CancelAllOrders	CancelQuote
CancelOrder	CreateQuote
CancelReplaceOrder	GetOpenQuotes
GetOpenOrders	UpdateQuote
GetOrderFee	
GetOrderHistory	
GetOrderStatus	
ModifyOrder	
SendOrder	

Contents Common to Many API Calls

These items appear in many of the API calls. Rather than explain them in place, we explain them here.

Note: There is occasional variance in the naming, spelling, and capitalization of string names, even those string/value pairs that refer to the same thing. For example, *AssetId* and *ProductId* are not interchangeable, even though they refer to the same data. Naming, spelling, and capitalization must follow the forms shown in the document.

Order Types

Used by: **CancelReplaceOrder**, **GetOpenOrders**, **GetOpenQuotes**, **GetOrderFee**, **GetOrderHistory**, **GetOrderHistoryByOrderId**, **GetOrdersHistory**, **GetOrderStatus**, and **SendOrder**.

Where:

Type	Definition
0 Unknown	The order type is unknown. Because all orders have a type, this is an error condition.
1 Market	An order to buy or sell an instrument at the best available price. Contains no restrictions on price or time frame.
2 Limit	An order to buy or sell a set amount of an instrument at a specified price or better. A limit order may not be executed if the price set is not met during the time that the order is open.
3 StopMarket	An order to buy or sell only when an instrument reaches a set price. Once the instrument reaches this price, the order becomes a market order.
4 StopLimit	An order to buy or sell only when an instrument reaches a set price. Once the instrument reaches this price, the order becomes a limit order to buy or sell at the limit price or better.
5 TrailingStopMarket	An order that sets the stop price for an instrument at a price with a fixed offset relative to the market price. If the market moves and the stop price is reached, the order becomes a market order.
6 TrailingStopLimit	An order that recalculates the stop price for an instrument at a fixed offset relative to the market price. It also recalculates the limit price based on a different fixed offset. If the market reaches the stop price, the order becomes a limit order.
7 BlockTrade	A privately executed trade.

Display Quantity

Used by: **CancelReplaceOrder**, **GetOpenOrders**, **GetOpenQuotes**, **GetOrderHistory**, **GetOrderHistoryByOrderId**, **GetOrdersHistory**, **GetOrderStatus**, and **SendOrder**

Display Quantity is the quantity of a product available to buy or sell that is publicly displayed to the market. A larger quantity may be made available for buying or selling, but it may be disadvantageous to show that amount all at once.

The number of units in a *DisplayQuantity* field appears as that number until the total number of units available or sought drops below the *DisplayQuantity* value set by the user. For example, if there are 100 units offered, but the *DisplayQuantity* value is set to 10, 10 continues to display as trading continues, until the number of units available for sale drops below 10.

The default value is 1.

To make use of a *DisplayQuantity* value, an order must be a limit order with a reserve. See "Order Types" on page 7.

Time– and Date-Stamp Formats

Coinext software uses three different time- and date-stamp formats. Unless otherwise specified, POSIX format is used.

- **POSIXct** class stores date/time values as the number of seconds since 1 January 1970 (long integer). Coinext software often multiplies this number by 1000 for the number of milliseconds since 1 January 1970. For more information on this format, consult: https://www.stat.berkeley.edu/~s133/dates.html
- **ISO 8601** format stores the date and time with its time zone (in Coinext, that time zone is always Zulu or UTC time). For example:

  ```
  yyyymmddThhmmssZ
  20080915T155300Z
  ```

  Where T indicates the beginning of the time information, and Z (Zulu/UTC) indicates the time zone — in this case, Zulu time. For more information on this format, consult: http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/a003169814.htm
- **Microsoft ticks** format represents the number of ticks that have elapsed since 00:00:00 UTC, 1 January 0001, in the Gregorian calendar. A single tick represents one hundred nanoseconds (one ten-millionth of a second). There are 10,000 ticks in a millisecond; ten million ticks in a second. It does not include the number of ticks attributable to leap-seconds.

  Microsoft provides the following sample code in C# (https://msdn.microsoft.com/en-us/library/system.datetime.ticks(v=vs.110).aspx):

  ```
  DateTime centuryBegin = new DateTime(2001, 1,

  1); DateTime currentDate = DateTime.Now;

  long elapsedTicks = currentDate.Ticks - centuryBegin.Ticks;

  TimeSpan elapsedSpan = new TimeSpan(elapsedTicks);

  Console.WriteLine("Elapsed from the beginning of the century
  to {0:f}:",
                     currentDate);
  Console.WriteLine("    {0:N0} nanoseconds", elapsedTicks * 100);
  Console.WriteLine("    {0:N0} ticks", elapsedTicks);
  Console.WriteLine("    {0:N2} seconds", elapsedSpan.TotalSeconds);
  Console.WriteLine("    {0:N2} minutes", elapsedSpan.TotalMinutes);
  Console.WriteLine("    {0:N0} days, {1} hours, {2} minutes, {3}
  seconds",
                    elapsedSpan.Days, elapsedSpan.Hours,
                    elapsedSpan.Minutes, elapsedSpan.Seconds);
  // If run on December 14, 2007, at 15:23, this example displays the
  // following output to the console:
  //    Elapsed from the beginning of the century to Friday, December 14,
  2007 3:23 PM:
  //          219,338,580,000,000,000 nanoseconds
  ```

Code continued on page 9

Code continued from page 8

```
//          2,193,385,800,000,000 ticks
//          219,338,580.00 seconds
//          3,655,643.00 minutes
//          2,538 days, 15 hours, 23 minutes, 0 seconds
```

The Trading Day

Most Coinext installations operate 24-hour computer-based trading venues. The trading day runs from UTC Midnight to UTC Midnight (essentially, London UK time, but without a summer offset). For values that comprise a per-day quantity (*TotalDayDeposits*, for example), the day runs from UTC Midnight to UTC Midnight, regardless of the venue's nominal location.

Deposit and Withdraw Templates

Templates provide a set of information about banking tasks during deposits and withdrawals, in the form of specific string/value pairs. Each template has a name. There are different templates for different types of deposit and withdrawal, determined by the product or asset (BitCoin, Monero, US Dollar, etc.), the specific bank or other account provider, and the information that the account provider requires for transactions.

Most templates are used for withdrawals.

Following, are two example templates.

```
"TemplateFormType": "Standard",
{
  "Full Name": "John Smith",
  "Language": "en",
  "Comment" : "",
  "BankAddress": "123 Fourth St.",
  "BankAccountNumber": "12345678",
  "BankAccountName": "John Smith & Sons",
  "SwiftCode": "ABCDUSA1"
}
"TemplateFormType": "TetherRpcWithdraw",
{
  "TemplateType": "TetherRpcWithdraw",
  "Comment": "TestWithdraw",
  "ExternalAddress": "ms6C3pKAAr8gRCcnVebs8VRkVrjcvqNYv3"
}
```

The content of the template depends on the account provider that you use for deposits and withdrawals. The account provider does not supply the template *per se* (they do, however, determine the fields that are in the template). The template is specific to each account provider. In one case, an unusual requirement of the account provider necessitated in the pre-population of certain request fields.

To determine which withdrawal template types are available to you, call **GetWithdrawTemplateTypes.**

Report Types

There are three report types:

- **Trade Activity:** Generates a report on both open and executed trades made by a set of Account IDs on a given Order Management System during a specified period.
- **Transaction:** Generates a report on all transactions executed by a set of Account IDs on a given Order Management System during a specified period.
- **Treasury:** Generates a report on all company treasury activities related to the trading venue — withdrawals, transfers, and funds movements unrelated to trading. The report comprises activities by a set of Account IDs on the given Order Management System for a specified period.

The Order Management System echoes back the Report Type as a confirmation of the call.

Request Status

When you generate a report on demand or schedule a report to run with some periodicity, the return object for the call provides the status of the report request in the RequestStatus string/value pair.

In the case of a Generate or Schedule call, RequestStatus returns Submitted; in the case of a GetUserReportTickets call, RequestStatus returns the status of the report within the system.

Table 1. Request Status definitions

Type	Definition
0 Submitted	Your report order has been submitted to the system.
1 Validating	The system is making sure that you have the correct permissions to request the report. See “Permissions” on page 4.
2 Scheduled	The report is scheduled to be run.
3 InProgress	The report is currently being prepared.
4 Completed	The report has been completed and delivered.
5 Aborting	The system is stopping preparation of the report.
6 Aborted	The report preparation has stopped.
7 UserCanceled	You have canceled this report.
8 SysRetired	The system has canceled the report on your behalf.
9 UserCanceledPending	You have requested a report cancellation, but the report has not been canceled yet.

API calls that return requestStatus are: **GenerateTradeActivityReport**, **GenerateTransactionActivityReport**, **GenerateTreasuryActivityReport**, **GetUserReportTickets**, **ScheduleTradeActivityReport**, **ScheduleTreasuryActivityReport**, and **ScheduleTreasuryActivityReport**.

Authentication

authenticate

No authentication required

authenticate authenticates a user (logs in a user) for the current websocket session. You must call **authenticate** in order to use the calls in this document not otherwise shown as "No authentication required."

Request

Use the standard, basic HTTP authentication, sending the username and password. A curl command would be something like:

```
curl -v 'https://your%40email.com:your-password@api.coinext.com.br:8443/AP/authenticate'
```

Response

Unsuccessful response:

```
{
  "Authenticated": false
}
```

Where:

String	Value
Authenticated	**Boolean.** The default response is *false* for an unsuccessful authentication.

A successful response returns the following (with *UserId* and *SessionToken* simulated):

```
{
  "Authenticated": true,
  "Token":"7d0ccf3a-ae63-44f5-a409-2301d80228bc",
  "UserId": 1,
  "AccountId": 1,
  "OMSId: 1
}
```

Where:

User Object:

String	Value
Authenticated	**Boolean.** The response is *true* for a successful authentication.

Table continued on page 16

Table continued from page 15

SessionToken	**string.** *SessionToken* uniquely identifies the session on the OMS. By returning the SessionToken in the response, the user can log in again if the session is interrupted without going through two-factor authentication.
UserId	**integer.** Returns the user ID of the authenticated user.

See Also

Authenticate2FA

Authenticate2FA

No authentication required

Completes the second part of a two-factor authentication by sending the authentication token from the non-Coinext authentication system to the Order Management System. The call returns a verification that the user logging in has been authenticated, and a token.

Here is how the two-factor authentication process works:

1. Call **WebAuthenticateUser**. The response includes values for *TwoFAType* and *TwoFAToken*. For example, *TwoFAType* may return "Google," and the *TwoFAToken* then returns a Google-appropriate token (which in this case would be a QR code).
2. Enter the *TwoFAToken* into the two-factor authentication program, for example, Google Authenticator. The authentication program returns a different token.
3. Call **Authenticate2FA** with the token you received from the two-factor authentication program (shown as *YourCode* in the request example below).

Request

```
{
  "Code": "YourCode"
}
```

Where:

String	Value
Code	**string.** Code holds the token obtained from the other authentication source.

Response

```
{
 "Authenticated": true,
 "SessionToken": "YourSessionToken"
}
```

Where:

String	Value
Authenticated	**Boolean.** A successful authentication returns *true*. Unsuccessful returns *false*.
SessionToken	**string.** The *SessionToken* is valid during the current session for connections from the same IP address. If the connection is interrupted during the session, you can sign back in using the *SessionToken* instead of repeating the full two-factor authentication process. A session lasts one hour after last-detected activity or until logout.

To send a session token to re-establish an interrupted session, send:

```
{
  “SessionToken”: “YourSessionToken”
}
```

See Also

WebAuthenticateUser, **LogOut**

LogOut

No authentication required

Logout ends the current websocket session.

Request

There is no payload for a *Logout* request.

```
{ }
```

Response

```
{
  “result”:true,
  “errormsg”:null,
  “errorcode”:0,
  “detail”:null
}
```

Where:

String	Value
result	**Boolean.** A successful logout returns true; and unsuccessful logout (an error condition) returns false.
errormsg	**string.** A successful logout returns null; the errormsg parameter for an unsuccessful logout returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104) Not Authorized and Resource Not Found are unlikely errors for a LogOut.
errorcode	**integer.** A successful logout returns 0. An unsuccessful logout returns one of the errorcodes shown in the errormsg list.
detail	**string.** Message text that the system may send. Usually null.

See Also:

authenticate, **Authenticate2FA**

LogOut

User Information Calls

GetUserInfo

Retrieves basic information about a user from the Order Management System. A user may only see information about himself; an administrator (or superuser) may see, enter, or change information about other users. See “Permissions” on page 4.

Request

No UserId is required in the request. The system assumes the current use.

```
{ }
```

Response

A successful response displays the settings for the user. An unsuccessful response generates an error code. See “Standard Response Object and Common Error Codes” on page 2.

```
{
  “UserId”: 1,
  “UserName”: “John Smith”,
  “Email”: “email@company.com”,
  “PasswordHash”: “”,
  “PendingEmailCode”: “”,
  “EmailVerified”: true,
  “AccountId”: 1,
  “DateTimeCreated”:”2017-10-26T17:25:58Z”,
  “AffiliateId”: 1,
  “RefererId”: 1,
  “OMSId”: 1,
  “Use2FA”: false,
  “Salt”: “”,
  “PendingCodeTime”: “0001-01-01T00:00:00Z”,
}
```

Where:

String	Value
UserId	**integer.** ID number of the user whose information is being set.
UserName	**string.** Log-in name of the user; “jsmith”.
Email	**string.** Email address of the user; “person@company.com”.
PasswordHash	**string.** Not currently used. Returns an empty string.
PendingEmailCode	**string.** Usually contains an empty string. During the time that a new user has been sent a registration email and before the user clicks the confirmation link, this pair contains a GUID — a globally unique ID string..
EmailVerified	**Boolean.** Has your organization verified this email as correct and operational? *True* if yes; *false* if no. Defaults to *false*.
AccountId	**integer.** The ID of the default account with which the user is associated.

Table continued on page 30

Table continued from page 29

DateTimeCreated	**long integer.** The date and time at which this user record was created, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.
AffiliatedId	**integer.** The ID of an affiliated organization, if the user comes from an affiliated link. This is set to 0 if the user it not associated with an affiliated organization.
RefererId	**integer.** Captures the ID of the person who referred this account member to the trading venue, usually for marketing purposes. Returns 0 if no referrer.
OMSId	**integer.** The ID of the Order Management System with which the user is associated.
Use2FA	**Boolean.** *True* if the user must use two-factor authentication; *false* if the user does not need to use two-factor authentication. Defaults to *false*.
Salt	**string.** Reserved for future use. Currently returns an empty string.
PendingCodeTime	**long integer.** A date and time in ISO 8601 format. Reserved. See “Time– and Date-Stamp Formats” on page 8.

See Also

GetAvailablePermissionList, GetUserPermissions, RegisterNewUser, SetUserConfig, SetUserInfo

Order-handling calls

CancelAllOrders

Cancels all open matching orders for the specified instrument, account, user (subject to permission level) or a combination of them on a specific Order Management System. User and account permissions govern cancellation actions. See "Permissions" on page 4. For more information on quotes and orders, see the explanation of "Quotes and Orders" on page 5.

Note: Multiple users may have access to the same account.

Specifying this information…			Cancels all orders for…
User 37	Acc't 14	Instr 25	
X	X	X	Account #14 belonging to user #37 for instrument #25.
X	X		Account #14 belonging to user #37 for all instruments.
X		X	All accounts belonging to user #37 for instrument #25.
X			All accounts belonging to user #37 for all instruments.
	X	X	All users of account #14 for instrument #25.
	X		All users of account #14 for all instruments.
		X	All accounts of all users for instrument #25. (requires special permission)
			All accounts of all users for all instruments (requires special permission)

Request

```
{
  "AccountId": 0, // conditionally optional
  "UserId": 0, // conditionally optional
  "OMSId": 0
  "InstrumentId": 0, // conditionally optional
}
```

Where:

String	Value
AccountId	**integer.** The account for which all orders are being canceled. Conditionally optional.
UserId	**integer.** The ID of the user whose orders are being canceled. Conditionally optional.
OMSId	**integer.** The Order Management System under which the account operates. Required.

Table continued on page42

Table continued from page 41

InstrumentId	**long integer.** The ID of the instrument for which all orders are being cancelled. Conditionally optional.

Response

The response to **CancelAllOrders** verifies that the call was received, not that the orders have been canceled successfully. Individual event updates to the user show orders as they cancel. To verify that an order has been canceled, use **GetOrderStatus** or **GetOpenOrders**. :

```
{
  “result”: true,
  “errormsg”: “”,
  “errorcode”: 0,
  “detail”: “”,
}
```

Where:

String	Value
result	**Boolean.** If the call has been successfully received by the Order Management System, result is *true*; otherwise, it is *false*.
errormsg	**string.** A successful receipt of the call returns null; the *errormsg* parameter for an unsuccessful call returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the call returns 0. An unsuccessful receipt of the call returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. The content of this parameter is usually *null*.

See Also

CancelOrder, CancelQuote, CancelReplaceOrder, CreateQuote, GetOpenOrders, GetOpenQuotes, GetOrderStatus, ModifyOrder, SendOrder, UpdateQuote

CancelOrder

Cancels an open order that has been placed but has not yet been executed. Only a trading venue operator can cancel orders for another user or account. See the explanation of ""Quotes and Orders" on page 5.

Request

The OMS ID and the Order ID precisely identify the order you wish to cancel. The Order ID is unique across an OMS.

If you specify the OMS ID and the Account ID, you must also specify at least the Client Order ID. The OMS is unable to identify the order using only the OMS ID and the Client Order ID, as the Client Order ID may not be unique.

```
{
  "OMSId": 0,
  "AccountId": 0       // conditionally optional
  "ClientOrderId": 0   // conditionally optional
  "OrderId": 0,        // conditionally optional
}
```

Where:

String	Value
OMSId	**integer.** The Order Management System on which the order exists. Required.
AccountId	**integer.** The ID of account under which the order was placed. Conditionally optional.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). ClientOrderId defaults to 0. Conditionally optional.
OrderId	**long integer.** The order to be cancelled. Conditionally optional.

Response

The response to **CancelOrder** verifies that the call was received, not that the order has been canceled successfully. Individual event updates to the user show order cancellation. To verify that an order has been canceled, call **GetOrderStatus** or **GetOpenOrders**. :

```
{
  "result": true,
  "errormsg": "",
  "errorcode": 0,
  "detail": "",
}
```

Where:

String	Value
result	**Boolean.** Returns true if the call to cancel the order has been successfully received, otherwise returns false.
errormsg	**string.** A successful receipt of a call to cancel an order returns null; the errormsg parameter for an unsuccessful call to cancel an order returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successfully received call to cancel an order returns 0. An unsuccessfully recieved call to cancel an order returns one of the errorcodes shown in the errormsg list.
detail	**string.** Message text that the system may send. The contents of this parameter are usually null.

See Also

CancelAllOrders, CancelQuote, CancelReplaceOrder, CreateQuote, GetOpenOrders, GetOpenQuotes, GetOrderStatus, ModifyOrder, SendOrder, UpdateQuote

CancelQuote

Cancels a quote that has not been executed yet.

Quoting is not enabled for the retail end user of the Coinext software. Only registered market participants or market makers may quote. Only a trading venue operator can cancel quotes for another user. See the explanation of "Quotes and Orders" on page 5.

Request

You must identify the quote to be canceled by both *BidQuoteId* and *AskQuoteId*, which were supplied by the system when the quote was created. You can optionally identify the canceled quote using *AccountId* and *InstrumentId*. If the call does not include *AccountId*, the call assumes the default *AccountId* for the logged-in user; if the call does not include *InstrumentId*, the call operates on any instruments quoted by the account.

```
{
  "OMSId": 0,
  "AccountId": 0, // conditionally optional
  "InstrumentId": 0, // conditionally optional
  "BidQuoteId": 0, // required
  "AskQuoteId": 0, // required
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the quote was requested. Required.
AccountId	**integer.** The ID of the account that requested the quote. Conditionally optional.
InstrumentId	**long integer.** The ID of the instrument being quoted. Conditionally optional.
BidQuoteId	integer. The ID of the bid quote. Required.
AskQuoteId	**integer.** The ID of the ask quote. Required.

Response

Returns two response objects, one for Bid and one for Ask.

The response to **CancelQuote** verifies that the call was received, not that the quote has been canceled successfully. Individual event updates to the user show quotes as they cancel. To verify that a quote has been canceled, use **GetOpenQuotes**.

```
{
  "bidresult": "{
    "result": true,
    "errormsg": "",
    "errorcode": 0,
    "detail": "",
  }",
  askresult": "{
    "result": true,
```

Code continued on page 46

Code continued from page 45

```
                    “errormsg”: “”,
                    “errorcode”: 0,
                    “detail”: “”,
                  }”
                }
```

Where:

String	Value
BidResult	**object.** Returns a standard response object for Bid (see below).
AskResult	**object.** Returns a standard response object for Ask.

Response objects for both *BidResult* and *AskResult*:

String	Value
result	**Boolean.** A successful receipt of the cancellation returns true; and unsuccessful receipt of the cancellation (an error condition) returns false.
errormsg	**string.** A successful receipt of the cancellation returns null; the errormsg parameter for an unsuccessful receipt returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the cancellation returns 0. An unsuccessful receipt returns one of the errorcodes shown in the errormsg list.
detail	**string.** Message text that the system may send. Usually null.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, CreateQuote, GetOpenOrders, GetOpenQuotes, GetOrderStatus, ModifyOrder, SendOrder, UpdateQuote

CancelReplaceOrder

CancelReplaceOrder is single API call that both cancels an existing order and replaces it with a new order. Canceling one order and replacing it with another also cancels the order's priority in the order book. You can use **ModifyOrder** to preserve priority in the book; but **ModifyOrder** only allows a reduction in order quantity.

Note: **CancelReplaceOrder** sacrifices the order's priority in the order book.

Request

```
{
  "OMSId": 0,
  "OrderIdToReplace": 0,
  "ClientOrdId": 0,
  "OrderType": {
    "Options": [
      "Unknown",
      "Market",
      "Limit",
      "StopMarket",
      "StopLimit",
      "TrailingStopMarket",
      "TrailingStopLimit",
      "BlockTrade"
    ]
  },
  "Side": {
    "Options":  [
      "Buy",
      "Sell",
      "Short",
      "Unknown",
    ]
  },
  "AccountId": 0,
  "InstrumentId": 0,
  "TrailingAmount": 0,
  "LimitOffset": 0,
  "DisplayQuantity": 0,
  "LimitPrice": 0,
  "StopPrice": 0, // conditionally optional
  "PegPriceType": {
   "Options":  [
      "Unknown",
      "Last",
      "Bid",
      "Ask",
      "Midpoint"
    ]
  },
  "TimeInForce": {
    "Options":  [
      "Unknown",
      "GTC",
      "IOC",
      "FOK",
    ]
  },
```

Code continues on page 48

Code continued from page 47

```
    "OrderIdOCO": 0,
    "Quantity": 0,
  }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the order is being canceled and replaced by another order.
OrderIdToReplace	**long integer.** The ID of the order to replace with this order.
ClientOrderId	**long integer.** A user-assigned ID for the new, replacement order (like a purchase-order number assigned by a company). This ID is useful for recognizing future states related to this order. *ClientOrderId* defaults to 0.
OrderType	**string.** The type of the replacement order: See Order Types in "Contents common to many API calls. 0 Unknown 1 Market 2 Limit 3 StopMarket 4 StopLimit 5 TrailingStopMarket 6 TrailingStopLimit 7 BlockTrade
Side	**string.** The side of the replacement order: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
AccountId	**integer.** The ID of the account under which the original order was placed and the new order will be placed.
InstrumentId	**integer.** The ID of the instrument being traded.
TrailingAmount	**real.** The offset by which to trail the market in one of the trailing order types. Set this to the current price of the market to ensure that the trailing offset is the amount intended in a fast-moving market.
LimitPrice	**real.** The price at which to execute the new order, if the order is a Limit order.
StopPrice	**real.** The price at which to execute the new order, if the order is a Stop order (either buy or sell).
PegPriceType	**string.** When entering a stop/trailing order, set *PegPriceType* to the type of price that pegs the stop. 1 Last 2 Bid 3 Ask 4 Midpoint

Table continued on page 49

Table continued from page 48

TimeInForce	**string.** The period during which the new order is executable. 0 Unknown (error condition) 1 GTC good 'til canceled 3 IOC immediate or canceled 4 FOK fill or kill — fill the order immediately, or cancel it immediately There may be other settings for TimeInForce depending on the trading venue.
OrderIdOCO	**integer.** One Cancels the Other — If the order being canceled in this call is order A, and the order replacing order A in this call is order B, then *OrderIdOCO* refers to an order C that is currently open. If order C executes, then order B is canceled. You can also set up order C to watch order B in this way, but that will require an update to order C.
Quantity	**real.** The amount of the order (buy or sell).

Response

The response returns the new replacement order ID and echoes back any replacement client ID you have supplied, along with the original order ID and the original client order ID.

```
{
  "ReplacementOrderId": 1234,
  "ReplacementClOrdId": 1561,
  "OrigOrderId": 5678,
  "OrigClOrdId": 91011,
}
```

Where:

String	Value
ReplacementOrderId	**integer.** The order ID assigned to the replacement order by the server.
ReplacementClOrdId	**long integer.** Echoes the contents of the *ClientOrderId* value from the request.
OrigOrderId	**integer.** Echoes *OrderIdToReplace*, which is the original order you are replacing.
OrigClOrdId	**long integer.** Provides the client order ID of the original order (not specified in the requesting call).

See Also

CancelAllOrders, CancelOrder, CancelQuote, CreateQuote, GetOpenOrders, GetOpenQuotes, GetOrderStatus, ModifyOrder, SendOrder, UpdateQuote

CreateQuote

Creates a quote. A quote expresses a willingness to buy or sell at a given price. See “Quotes and Orders” on page 5 for a discussion of how quotes and orders differ. Both a quote and an order will execute. Quoting is not enabled for the retail end user of Coinext software. Only registered market participants or market makers may quote.

Request

```
{
  “OMSId”: 0,
  “AccountId”: 0,
  “InstrumentId”: 0,
  “Bid”: 0,
  “BidQty”: 0,
  “Ask”: 0,
  “AskQty”: 0,
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the quote is being created. Required.
AccountId	**integer.** The ID of the account in which the quote is being created. If the call provides no *AccountId*, the system assumes the default account ID for the logged-in user on the OMS.
InstrumentId	**long integer.** The ID of the instrument being quoted. Required.
Bid	**real.** The bid price. Required.
BidQty	**real.** The quantity of the bid. Required.
Ask	**real.** The ask price. Required.
AskQty	**real.** The quantity of the ask. Required.

Response

```
{
  “BidQuoteId”: 0,
  “BidResult”: “{
    “result”: true,
    “errormsg”: “”,
    “errorcode”: 0,
    “detail”: “”,
  }”,
  “AskQuoteId”: 0,
  “AskResult”: “{
    “result”: true,
```

Code continued on page 52

Code continued from page 51

```
                "errormsg": "",
                "errorcode": 0,
                "detail": "",
              }"
            }
```

Where:

String	Value
BidQuoteId	**integer.** The ID of the bid quote returned by the Order Management System.
BidResult	**string.** Returns a standard response object for Bid.
AskQuoteId	integer. The ID of the ask quote returned by the Order Management System.
AskResult	**string.** Returns a standard response object for Ask.

Response objects for both *BidResult* and *AskResult*.

String	Value
result	**Boolean.** A successful receipt of the request to create a quote returns true; and unsuccessful receipt of the request (an error condition) returns false.
errormsg	**string.** A successful receipt of the request returns null; the *errormsg* parameter for an unsuccessful receipt returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the cancellation returns 0. An unsuccessful receipt returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

CancelQuote, GetOpenQuotes, UpdateQuote

GetAccountPositions

Retrieves a list of positions (balances) for a specific user account running under a specific Order Management System. The trading day runs from UTC Midnight to UTC Midnight. See "The Trading Day" on page 9 for more information.

Request

```
{
 "AccountId":4,
 "OMSId": 1
}
```

Where:

String	Value
AccountId	**integer.** The ID of the authenticated user's account on the Order Management System for which positions will be returned.
OMSId	**integer.** The ID of the Order Management System to which the user belongs. A user will belong only to one OMS.

Response

The response returns an array of one or more positions for the account. This example response has returned two positions:

```
[
  { // first position
    "OMSId":1,
    "AccountId":4,
    "ProductSymbol":"BTC"
    "ProductId":1
    "Amount":0,
    "Hold":0,
    "PendingDeposits":0,
    "PendingWithdraws":0,
    "TotalDayDeposits":0,
    "TotalDayWithdraws":0,
    "TotalMonthWithdraws":0
  },
  { //second position
    "OMSId":1,
    "AccountId":4,
    "ProductSymbol":"USD",
    "ProductId":2,
    "Amount":0, "Hold":0,
    "PendingDeposits":0,

    "PendingWithdraws":0,
    "TotalDayDeposits":0,
    "TotalDayWithdraws":0,
    "TotalMonthWithdraws":0
  }
]
```

Where:

String	Value
OMSId	**Integer.** The ID of the Order Management System (OMS) to which the user belongs. A user will only ever belong to one Order Management System.
AccountId	**integer.** Returns the ID of the user's account to which the positions belong.
ProductSymbol	**string.** The symbol of the product on this account's side of the trade. For example: BTC — BitCoin USD — US Dollar NZD — New Zealand Dollar Many other values are possible depending on the nature of the trading venue. See "Products and Instruments" on page 4 for the difference between these terms.
ProductId	**integer.** The ID of the product being traded. The system assigns product IDs as they are entered into the system. See "Products and Instruments" on page 4 for the difference between products and instruments. Use **GetProduct** to return information about the product by its ID.
Amount	**real.** Unit amount of the product; for example, 10 or 138.5.
Hold	**real.** Amount of currency held and not available for trade. A pending trade of 100 units at $1 each will reduce the amount in the account available for trading by $100. Amounts on hold cannot be withdrawn while a trade is pending.
PendingDeposits	**real.** Deposits accepted but not yet cleared for trade.
PendingWithdraws	**real.** Withdrawals acknowledged but not yet cleared from the account. Amounts in *PendingWithdraws* are not available for trade.
TotalDayDeposits	**real.** Total deposits on today's date. The trading day runs between UTC Midnight and UTC Midnight.
TotalDayWithdraws	**real.** Total withdrawals on today's date. The trading day runs between UTC Midnight and UTC Midnight.
TotalMonthWithdraws	**real.** Total withdrawals during this month to date. The trading day runs between UTC Midnight and UTC Midnight — likewise a month begins at UTC Midnight on the first day of the month.

See Also

GetOpenOrders, GetOpenQuotes, GetOrderStatus, GetTradesHistory

GetAccountTrades

Requests the details on up to 200 past trade executions for a single specific user account and its Order Management System, starting at index *i*, where *i* is an integer identifying a specific execution in reverse order; that is, the most recent execution has an index of 0, and increments by one as trade executions recede into the past.

The operator of the trading venue determines how long to retain an accessible trading history before archiving.

Request

```
{
  "AccountId":4,
  "OMSId": 1,
  "StartIndex":0,
  "Count":2
}
```

Where:

String	Value
AccountId	**integer.** The ID of the authenticated user's account.
OMSId	**integer.** The ID of the Order Management System to which the user belongs. A user will belong only to one OMS.
StartIndex	**integer.** The starting index into the history of trades, from 0 (the most recent trade).
Count	**integer.** The number of trades to return. The system can return up to 200 trades.

Response

The response is an array of objects, each of which represents the account's side of a trade (either buy or sell). The example shows an array of two buy executions.

```
[
  {
    "TradeTimeMS": -62135446664520,
    "Fee": 0,
    "FeeProductId": 0,
    "OrderOriginator": 1,
    "OMSId": 1,
    "ExecutionId": 1,
    "TradeId": 1,
    "OrderId": 1,
    "AccountId": 4,
    "SubAccountId": 0,
    "ClientOrderId": 0,
    "InstrumentId": 1,
    "Side": "Buy",
    "Quantity": 1,
    "RemainingQuantity": 0,
    "Price": 100,
```

Code continued on page 60

Code continued from page 59

```
      "Value": 100,
      "TradeTime": 1501354796406,
      "CounterParty": null,
      "OrderTradeRevision": 1,
      "Direction": "NoChange",
      "IsBlockTrade": false
    },
    {
      "TradeTimeMS": -62135446664520,
      "Fee": 0,
      "FeeProductId": 0,
      "OrderOriginator": 1,
      "OMSId": 1,
      "ExecutionId": 3,
      "TradeId": 2,
      "OrderId": 3,
      "AccountId": 4,
      "SubAccountId": 0,
      "ClientOrderId": 0,
      "InstrumentId": 1,
      "Side": "Buy",
      "Quantity": 1,
      "RemainingQuantity": 0,
      "Price": 1,
      "Value": 1,
      "TradeTime": 1501354796418,
      "CounterParty": null,
      "OrderTradeRevision": 1,
      "Direction": "NoChange",
      "IsBlockTrade": false
    }
  ]
```

Where:

String	Value
TradeTimeMS	**long integer.** The date and time stamp of the trade in Microsoft tick format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
Fee	**real.** The fee for this trade in units and fractions of units (a $10 USD fee would be 10.00, a .5-BitCoin fee would be 0.5).
FeeProductId	**integer.** The ID of the product that denominates the fee. Product types will vary on each trading venue. See **GetProduct**.
OrderOriginator	**integer.** The user ID of the user who entered the order that caused the trade for this account. (Multiple users can have access to an account.)
OMSId	**integer.** The ID of the Order Management System to which the user belongs. A user will belong only to one OMS.
ExecutionId	**integer.** The ID of this account's side of the trade. Every trade has two sides.
TradeId	**integer.** The ID of the overall trade.
OrderId	**long integer.** The ID of the order causing the trade.
AccountId	**integer.** The Account ID that made the trade.
SubAccountId	**integer.** Not currently used.

Table continued on page 61

Table continued from page 60

InstrumentId	**long integer.** The ID of the instrument being traded. See **“Products and Instruments” on page 4** for the difference. See **GetInstrument** to find information about this instrument by its ID.
Side	**string.** Buy or Sell 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
Quantity	**real.** The unit quantity of the trade.
RemainingQuantity	**integer.** The number of units remaining to be traded by the order after this execution. This number is not revealed to the other party in the trade. This value is also known as “leave size” or “leave quantity.”
Price	**real.** The unit price at which the instrument traded.
Value	**real.** The total value of the deal. The system calculates this as: unit price X quantity executed.
TradeTime	**integer.** The time at which the trade took place, in POSIX format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
CounterParty	**long integer.** Shows 0.
OrderTradeRevision	**integer.** This value increments if the trade has changed. Default is 1. For example, if the trade busts (fails to conclude), the trade will need to be modified and a revision number then will apply.
Direction	**string.** Shows if this trade has moved the book price up, down, or no change. Values: NoChange UpTick DownTick
IsBlockTrade	**Boolean.** Returns true if the trade was a reported trade; false otherwise.

See Also

GenerateTradeActivityReport, GetTradesHistory, ScheduleTradeActivityReport, SubscribeTrades, UnsubscribeTrades

GetAccountTrades

GetAccountTransactions

Returns a list of transactions for a specific account on an Order Management System. The owner of the trading venue determines how long to retain order history before archiving.

Note: In this call, “Depth” refers not to the depth of the order book, but to the count of trades to report.

Request

```
{
 “OMSId”:  1,
 “AccountId”:  1,
 “Depth”:  200
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System from which the account’s transactions will be returned.
AccountId	**integer.** The ID of the account for which transactions will be returned. If not specified, the call returns transactions for the default account for the logged-in user.
Depth	**integer.** The number of transactions that will be returned, starting with the most recent transaction.

Response

The response returns an array of transaction objects.

```
[
  {
    {
      “TransactionId”: 0,
      “OMSId”: 0,
      “AccountId”: 0,
      “CR”: 0,
      “DR”: 0,
      “Counterparty”: 0,
      “TransactionType”: {
         “Options”: [
          “Fee”,
          “Trade”,
          “Other”,
          “Reverse”,
          “Hold”
        ]
      },
      “ReferenceId”: 0,
      “ReferenceType”: {
         “Options”: [
          “Trade”,
          “Deposit”,
```

Code continues on page 64

Code continued from page 63

```
                "Withdraw",
                "Transfer",
                "OrderHold",
                "WithdrawHold",
                "DepositHold",
                "MarginHold"
              ]
            },
            "ProductId": 0,
            "Balance": 0,
            "TimeStamp": 0,
          },
        }
      ]
```

Where:

String	Value
TransactionId	**Integer.** The ID of the transaction.
OMSId	**Integer.** The ID of the Order Management System under which the requested transactions took place.
AccountId	**Integer.** The single account under which the transactions took place.
CR	**real.** Credit entry for the account on the order book. Funds entering an account.
DR	**real.** Debit entry for the account on the order book. Funds leaving an account.
Counterparty	**long integer.** Shows 0.
TransactionType	**string.** One of: Fee — transaction is payment of a fee Trade — transaction is a trade (most usual entry) Other — non-trading transactions such as deposits and withdrawals Reverse — a hold has been reversed by this transaction Hold — funds are held while a transaction closes
ReferenceId	**long integer.** The ID of the action or event that triggered this transaction.
ReferenceType	**string.** The type of action or event that triggered this transaction. One of: Trade Deposit Withdraw Transfer OrderHold WithdrawHold DepositHold MarginHold
ProductId	**integer.** The ID of the product on this account's side of the transaction. For example, in a dollars-for-BitCoin transaction, one side will have the product Dollar and the other side will have the product BitCoin. See "Products and Instruments" on page 4 for more information about how these two items differ. Use **GetProduct** to return information about a product based on its ID.
Balance	**real.** The balance in the account after the transaction.
TimeStamp	**long integer.** Time at which the transaction took place, in POSIX format and UTC time zone.

See Also

GetAccountTransactions, ScheduleTransactionActivityReport

GetInstrument

No authentication required

Retrieves the details of a specific instrument from the Order Management System of the trading venue. An instrument is a pair of exchanged products (or fractions of them) such as US dollars and ounces of gold. See “Products and Instruments” on page 4 for more information about how products and instruments differ.

Request

```
{
 “OMSId”: 1,
 “InstrumentId”: 1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System from where the instrument is traded.
InstrumentId	**long integer.** The ID of the instrument.

Response

```
{
  “OMSId”: 0,
  “InstrumentId”: 0,
  “Symbol”: “”,
  “Product1”: 0,
  “Product1Symbol”: “”,
  “Product2”: 0,
  “Product2Symbol”: “”,
  “InstrumentType”: {
    “Options”:  [
      “Unknown”,
      “Standard”
    ]
  },
  “VenueInstrumentId”: 0,
  “VenueId”: 0,
  “SortIndex”: 0,
  “SessionStatus”: {
    “Options”:  [
      “Unknown”,
      “Running”,
      “Paused”,
      “Stopped”,
      “Starting”
    ]
  },
  “PreviousSessionStatus”: {
    “Options”:  [
      “Unknown”,
      “Running”,
      “Paused”,
```

Code continued on page 68

Code continued from page 67

```
                "Stopped",
                "Starting"
              ]
            },
            "SessionStatusDateTime": "0001-01-01T05:00:00Z",
            "SelfTradePrevention": false,
            "QuantityIncrement": 0,

          }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instrument is traded.
InstrumentId	**long integer.** The ID of the instrument.
Symbol	**string.** Trading symbol of the instrument.
Product1	**integer.** The first product comprising the instrument. For example, USD in a USD/ BitCoin instrument.
Product1Symbol	**string.** The symbol for Product 1 on the trading venue. For example, USD.
Product2	**integer.** The second product comprising the instrument. For example, BitCoin in a USD/BitCoin instrument.
Product2Symbol	**string.** The symbol for Product 1 on the trading venue. For example, BTC.
InstrumentType	**string.** The type of the instrument. All instrument types currently are *standard*, an exchange of one product for another (or unknown, an error condition), but this may expand to new types in the future. Unknown Standard
VenueInstrumentId	**long integer.** If the instrument trades on another trading venue to which the user has access, this value is the instrument ID on that other venue. See *VenueId*.
VenueId	**integer.** The ID of the trading venue on which the instrument trades, if not this venue. See *VenueInstrumentId*.
SortIndex	**integer.** The numerical position in which to sort the returned list of instruments on a visual display. Since this call returns information about a single instrument, *SortIndex* should return 0.
SessionStatus	**string.** Is the market for this instrument currently open and operational? Returns one of: Unknown Running Paused Stopped Starting
PreviousSessionStatus	**string.** What was the previous session status for this instrument? One of: Unknown Running Paused Stopped Starting

Table continued on page 69

Table continued from page 68

SessionStatusDateTime	**string.** The time and date at which the session status was reported, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.
SelfTradePrevention	**Boolean.** An account trading with itself still incurs fees. If this instrument prevents an account from trading the instrument with itself, the value returns *true*; otherwise defaults to *false*.
QuantityIncrement	**integer.** The number of decimal places for the smallest quantity of the instrument that can trade (analogous to smallest lot size). For example, the smallest increment of a US Dollar that can trade is 0.01 (one cent, or 2 decimal places). Current maximum is 8 decimal places. The default is 0.

See Also

GetInstruments, GetProduct, GetProducts

GetInstruments

No authentication required

Retrieves an array of instrument objects describing all instruments available on a trading venue to the user. An instrument is a pair of exchanged products (or fractions of them) such as US dollars and ounces of gold. See “Products and Instruments” on page 4 for more information about how products and instruments differ.

Request

```
{
  “OMSId”:  1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instruments are available.

Response

The response for **GetInstruments** is an array of objects describing all the instruments available to the authenticated user on the Order Management System.

```
[
  {
    {
    “OMSId”: 0,
    “InstrumentId”: 0,
    “Symbol”: “”,
    “Product1”: 0,
    “Product1Symbol”: “”,
    “Product2”: 0,
    “Product2Symbol”: “”,
    “InstrumentType”: {
      “Options”:       [
        “Unknown”,
        “Standard”
        ]
      },
    “VenueInstrumentId”: 0,
    “VenueId”: 0,
    “SortIndex”: 0,
    “SessionStatus”: {
      “Options”:       [
        “Unknown”,
        “Running”,
        “Paused”,
        “Stopped”,
        “Starting”
        ]
```

Code continued on page 72

Code continued from page 71

```
                  },
                "PreviousSessionStatus": {
                  "Options":       [
                    "Unknown",
                    "Running",
                    "Paused",
                    "Stopped",
                    "Starting"
                    ]
                  },
                  "SessionStatusDateTime": "0001-01-01T05:00:00Z",
                  "SelfTradePrevention": false,
                  "QuantityIncrement": 0,
                },
              }
            ]
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instrument is traded.
InstrumentId	**long integer.** The ID of the instrument.
Symbol	**string.** Trading symbol of the instrument.
Product1	**integer.** The first product comprising the instrument. For example, USD in a USD/ BitCoin instrument.
Product1Symbol	**string.** The symbol for Product 1 on the trading venue. For example, USD.
Product2	**integer.** The second product comprising the instrument. For example, BitCoin in a USD/BitCoin instrument.
Product2Symbol	**string.** The symbol for Product 2 on the trading venue. For example, BTC.
InstrumentType	**string.** The type of the instrument. All instrument types currently are standard, an exchange of one product for another (or unknown, an error condition), but this may expand to new types in the future. Unknown Standard
VenueInstrumentId	**long integer.** If the instrument trades on another trading venue to which the user has access, this value is the instrument ID on that other venue. See *VenueId*.
VenueId	**integer.** The ID of the trading venue on which the instrument trades, if not this venue. See *VenueInstrumentId*.
SortIndex	**integer.** The numerical position in which to sort the returned list of instruments on a visual display.
SessionStatus	**string.** Is the market for this instrument currently open and operational? Returns one of: Unknown Running Paused Stopped Starting

Table continued on page 73

Table continued from page 72

PreviousSessionStatus	**string.** What was the previous session status for this instrument? One of: Unknown Running Paused Stopped Starting
SessionStatusDateTime	**string.** The time and date at which the session status was reported, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.
SelfTradePrevention	**Boolean.** An account trading with itself still incurs fees. If this instrument prevents an account from trading the instrument with itself, the value returns *true*; otherwise defaults to *false*.
QuantityIncrement	**integer.** The number of decimal places for the smallest quantity of the instrument that can trade (analogous to smallest lot size). For example, the smallest increment of a US Dollar that can trade is 0.01 (one cent, or 2 decimal places). Current maximum is 8 decimal places. The default is 0.

See Also

GetInstrument, GetProduct, GetProducts

GetOpenOrders

Returns an array of 0 or more orders that have not yet been filled (open orders) for a single account for a given user on a specific Order Management System. The call returns an empty array if a user has no open orders.

Request

```
{
 “AccountId”:4,
 “OMSId”: 1
}
```

Where:

String	Value
AccountId	**integer.** The ID of the authenticated user’s account.
OMSId	**integer.** The ID of the Order Management System to which the user belongs. A user will belong only to one OMS.

Response

This example response for GetOpenOrders returns an array containing both a buy-side and a sell-side order. The call returns an empty array if there are no open orders for the account.

```
[
  {
    “Side”: “Buy”,
    “OrderId”: 1,
    “Price”: 100,
    “Quantity”: 1,
    “DisplayQuantity”: 1,
    “Instrument”: 1,
    “Account”: 4,
    “OrderType”: “Limit”,
    “ClientOrderId”: 0,
    “OrderState”: “Working”,
    “ReceiveTime”: 1501354241987,
    “ReceiveTimeTicks”: 636369510419870950,
    “OrigQuantity”: 1,
    “QuantityExecuted”: 0,
    “AvgPrice”: 0,
    “CounterPartyId”: 0,
    “ChangeReason”: “NewInputAccepted”,
    “OrigOrderId”: 1,
    “OrigClOrdId”: 0,
    “EnteredBy”: 1,
    “IsQuote”: false,
    “InsideAsk”: 9223372036.854775807,
    “InsideAskSize”: 0,
    “InsideBid”: 100,
    “InsideBidSize”: 1,
    “LastTradePrice”: 0,
    “RejectReason”: “”,
    “IsLockedIn”: false,
```

Code continues on page 76

Code continued from page 75

```
    "OMSId": 1
  },
  {
    "Side": "Sell",
    "OrderId": 2,
    "Price": 150,
    "Quantity": 1,
    "DisplayQuantity": 1,
    "Instrument": 1,
    "Account": 4,
    "OrderType": "Limit",
    "ClientOrderId": 0,
    "OrderState": "Working",
    "ReceiveTime": 1501354246718,
    "ReceiveTimeTicks": 636369510467182396,
    "OrigQuantity": 1,
    "QuantityExecuted": 0,
    "AvgPrice": 0,
    "CounterPartyId": 0,
    "ChangeReason": "NewInputAccepted",
    "OrigOrderId": 2,
    "OrigClOrdId": 0,
    "EnteredBy": 1,
    "IsQuote": false,
    "InsideAsk": 150,
    "InsideAskSize": 1,
    "InsideBid": 100,
    "InsideBidSize": 1,
    "LastTradePrice": 0,
    "RejectReason": "",
    "IsLockedIn": false,
    "OMSId": 1
  }
]
```

String	Value
Side	**string.** The open order can be Buy or Sell. 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
OrderId	**long integer.** The ID of the open order. The *OrderID* is unique in each Order Management Systsem.
Price	**real.** The price at which the buy or sell has been ordered.
Quantity	**real.** The quantity to be bought or sold.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a *DisplayQuantity* value, an order must be a Limit order with a reserve. See "Display Quantity" on page 8, and "Order Types" on page 7.
Instrument	**integer.** ID of the instrument being traded. See **GetInstruments**.
Account	**integer.** The ID of the account that placed the order.
OrderType	**string.** There are currently seven types of order. See "Order Types" on page 7.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.

Table continued on page 77

Table continued from page 76

OrderState	**string.** The current condition of the order. There are five order states: Working Rejected Canceled Expired FullyExecuted
ReceiveTime	**long integer.** The time at which the system received the order, in POSIX format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
ReceiveTimeTicks	**long integer.** The time stamp of the received order in Microsoft Tick format, and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
OrigQuantity	**integer.** Original quantity of the order. The quantity of the actual execution may be lower than this number, but *OrigQuantity* shows the quantity in the order as placed.
QuantityExecuted	**Integer.** The number of units executed in this trade.
AvgPrice	**real.** Not currently used.
CounterPartyId	**long integer.** Shows 0.
ChangeReason	**string.** The reason that an order has been changed. Values: 1 NewInputAccepted 2 NewInputRejected 3 OtherRejected 4 Expired 5 Trade 6 SystemCanceled_NoMoreMarket 7 SystemCanceled_BelowMinimum 8 NoChange 100 UserModified
OrigOrderId	**long. ID of th**e original order. This number is also appended to **CancelReplaceOrder**. See **CancelReplaceOrder**.
EnteredBy	**integer.** User ID of the person who entered the order.
IsQuote	**Boolean.** *True* if the open order is a quote; *false* if not. See "Quotes and Orders" on page 5.
InsideAsk/InsideBid	**real.** Best price available at time of entry (for ask or bid, respectively).
InsideAskSize/ InsideBidSize	**real.** Quantity available at the best inside ask (or bid) price.
LastTradePrice	**real.** Last trade price for this product before this order was entered.
RejectReason	**string.** If this order was rejected, *RejectReason* holds the reason for the rejection.
IsLockedIn	**Boolean.** *True* if both parties to a block trade agree that one party will report the trade for both. Otherwise *false.*
OMSId	**integer.** ID of the Order Management System on which the order was placed.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOrdersHistory, GetOrderHistoryByOrderId, GetOrdersHistory, GetOrderStatus, ModifyOrder, SendOrder

GetOpenQuotes

Returns the current bid and ask quotes for a given instrument ID and account ID.

Request

```
{
  “OMSId”: 0,
  “AccountId”: 0,
  “InstrumentId”: 0,
 }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the instrument is traded whose quote may be open.
AccountId	**integer.** The ID of the account whose open quotes will be returned.
InstrumentId	**long integer.** The ID of the instrument being quoted.

Response

Returns a response object comprising a bid and an ask object.

```
{
  “Bid”: {
    “Side”: {
    “Options”: [
      “Buy”,
      “Sell”,
      “Short”,
      “Unknown”
      ]
    },
    “OrderId”: 0,
    “Price”: 0,
    “Quantity”: 0,
    “DisplayQuantity”: 0,
    “Instrument”: 0,
    “Account”: 0,
    “OrderType”: {
       “Options”: [
        “Unknown”,
        “Market”,
        “Limit”,
        “StopMarket”,
        “StopLimit”,
        “TrailingStopMarket”,
        “TrailingStopLimit”,
        “BlockTrade”
      ]
    },
    “ClientOrderId”: 0,
    “OrderState”: {
```

Code continued on page 80

Code continued from page 79

```
            "Options": [
             "Unknown",
             "Working",
             "Rejected",
             "Canceled",
             "Expired",
             "FullyExecuted"
           ]
         },
         "ReceiveTime": 0,
         "ReceiveTimeTicks": 0,
         "OrigQuantity": 0,
         "QuantityExecuted": 0,
         "AvgPrice": 0,
         "CounterPartyId": 0,
         "ChangeReason": {
            "Options": [
             "Unknown",
             "NewInputAccepted",
             "NewInputRejected",
             "OtherRejected",
             "Expired",
             "Trade",
             "SystemCanceled_NoMoreMarket",
             "SystemCanceled_BelowMinimum",
             "NoChange",
             "UserModified"
           ]
         },
         "OrigOrderId": 0,
         "OrigClOrdId": 0,
         "EnteredBy": 0,
         "IsQuote": false,
         "InsideAsk": 0,
         "InsideAskSize": 0,
         "InsideBid": 0,
         "InsideBidSize": 0,
         "LastTradePrice": 0,
         "RejectReason": "",
         "IsLockedIn": false,
         "OMSId": 0,
       },
       "Ask": {
         "Side": {
            "Options": [
             "Buy",
             "Sell",
             "Short",
             "Unknown"
           ]
         },
         "OrderId": 0,
         "Price": 0,
         "Quantity": 0,
         "DisplayQuantity": 0,
         "Instrument": 0,
         "Account": 0,
         "OrderType": {
            "Options": [
             "Unknown",
             "Market",
             "Limit",
             "StopMarket",
             "StopLimit",
             "TrailingStopMarket",
             "TrailingStopLimit",
             "BlockTrade"
           ]
         },
         "ClientOrderId": 0,
         "OrderState": {
```

Code continued on page 81

Code continued from page 80

```
                    "Options":  [
                     "Unknown",
                     "Working",
                     "Rejected",
                     "Canceled",
                     "Expired",
                     "FullyExecuted"
                   ]
                 },
                 "ReceiveTime": 0,
                 "ReceiveTimeTicks": 0,
                 "OrigQuantity": 0,
                 "QuantityExecuted": 0,
                 "AvgPrice": 0,
                 "CounterPartyId": 0,
                 "ChangeReason": {
                    "Options": [
                     "Unknown",
                     "NewInputAccepted",
                     "NewInputRejected",
                     "OtherRejected",
                     "Expired",
                     "Trade",
                     "SystemCanceled_NoMoreMarket",
                     "SystemCanceled_BelowMinimum",
                     "NoChange",
                     "UserModified"
                   ]
                 },
                 "OrigOrderId": 0,
                 "OrigClOrdId": 0,
                 "EnteredBy": 0,
                 "IsQuote": false,
                 "InsideAsk": 0,
                 "InsideAskSize": 0,
                 "InsideBid": 0,
                 "InsideBidSize": 0,
                 "LastTradePrice": 0,
                 "RejectReason": "",
                 "IsLockedIn": false,
                 "OMSId": 0,
               },
             }
```

Where:

String	Value
Bid	Bid object (see below)
Ask	Ask object (see below)

Bid and Ask objects differ only in the values for the strings.

String	Value
Side	**string.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)

Table continued on page 82

Table continued from page 81

OrderId	**long integer.** The ID of this quote. Quotes and orders are both executable. See “Quotes and Orders” on page 5.
Price	**real.** Price of the Bid/Ask quote.
Quantity	**real.** Quantity of the Bid/Ask quote.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a *DisplayQuantity* value, an order must be a Limit order with a reserve. See “Display Quantity” on page 8, and “Order Types” on page 7.
Instrument	**integer.** The ID of the instrument being quoted.
Account	**integer.** The ID of the account quoting the instrument.
OrderType	**string.** One of: Unknown Market Limit StopMarket StopLimit TrailingStopMarket TrailingStopLimit BlockTrade See “Order Types” on page 7.
ClientOrderId	**long integer.** A user-assigned ID for the quote (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
OrderState	**string.** One of: Unknown Working Rejected Canceled Expired FullyExecuted An open quote will probably have an *OrderState* of Working.
ReceiveTime	**long integer.** The time at which the system received the quote, in POSIX format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
ReceiveTimeTicks	**long integer.** The time stamp of the received quote in Microsoft Ticks format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
OrigQuantity	**real.** If the quote has been changed, this value shows the original quantity of the quote.
QuantityExecuted	**real.** This value states the quantity that was executed. It may be the same as the quantity of the quote; it may be different.
AvgPrice	**real.** Not currently used.
CounterPartyId	**long integer.** Shows 0.

Table continued on page 83

Table continued from page 82

ChangeReason	**string.** If the quote has been changed, this value shows the reason. One of: Unknown NewInputAccepted NewInputRejected OtherRejected Expired Trade SystemCanceled_NoMoreMarket SystemCanceled_BelowMinimum NoChange UserModified
OrigOrderId	**integer.** If the quote has been changed, shows the original order ID. (Quotes and orders are in some ways interchangeable. See “Quotes and Orders” on page 5.
OrigClOrdId	**long integer.** If the quote has been changed, shows the original client order ID, a value that the client can create (much like a purchase order).
EnteredBy	**integer.** The ID of the user who entered the quote.
IsQuote	**Boolean.** If this order is a quote (rather than an order), returns *true*, otherwise *false*. Default is *false*.
InsideAsk	**real.** Best Ask price available at time of entry (generally available to market makers).
InsideAskSize	**real.** Quantity available at the best inside ask price (generally available to market makers).
InsideBid	**real.** Best Bid price available at time of entry (generally available to market makers).
InsideBidSize	**real.** Quantity available at the best inside Bid price (generally available to market makers)..
LastTradePrice	**real.** The price at which the instrument last traded.
RejectReason	**string.** If the quote was rejected, this string value holds the reason.
IsLockedIn	**Boolean.** *True* if both parties to a block trade agree that one party will report the trade for both. Otherwise *false*.
OMSId	**integer.** The ID of the Order Management System on which the quote was created.

See Also

CancelQuote, CreateQuote, UpdateQuote

GetOrderFee

Returns an estimate of the fee for a specific order and order type. Fees are set and calculated by the operator of the trading venue.

Request

```
{
  "OMSId": 0,
  "AccountId": 0,
  "InstrumentId": 0,
  "ProductId": 0,
  "Amount": 0,
  "Price": 0,
  "OrderType": {
     "Options": [
      "Unknown",
      "Market",
      "Limit",
      "StopMarket",
      "StopLimit",
      "TrailingStopMarket",
      "TrailingStopLimit",
      "BlockTrade"
    ]
  },
  "MakerTaker": {
     "Options": [
      "Unknown",
      "Maker",
      "Taker"
     ]
   },
 }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the trade would take place.
AccountId	**integer.** The ID of the account requesting the fee estimate.
InstrumentId	**integer.** The proposed instrument against which a trading fee would be charged.
ProductId	**integer.** The ID of the product (currency) in which the fee will be denominated.
Amount	**real.** The quantity of the proposed trade for which the Order Management System would charge a fee.
Price	**real.** The price at which the proposed trade would take place. Supply your price for a limit order; the exact price is difficult to know before execution.

Table continued on page 86

Table continued from page 85

OrderType	**string.** The type of the proposed order. One of: 0 Unknown 1 Market 2 Limit 3 StopMarket 4 StopLimit 5 TrailingStopMarket 6 TrailingStopLimit 7 BlockTrade See ““Order Types” on page 7.
MakerTaker	**string.** Depending on the venue, there may be different fees for a maker (the order remains on the books for a period) or taker (the order executes directly). If the user places a large order that is only partially filled, he is a partial maker. 0 Unknown 1 Maker 2 Taker

Response

```
{
  "OrderFee":  0.01,
  "ProductId":   1
}
```

Where:

String	Value
OrderFee	**real.** The estimated fee for the trade as described. The minimum value is 0.01.
ProductId	**integer.** The ID of the product (currency) in which the fee is denominated.

See Also

GetProduct, GetProducts

GetOrderHistory

Returns a complete list of all orders, both open and executed, for a specific account on the specified Order Management System.

Request

```
{
  "OMSId":  1,
  "AccountId":  1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the orders were placed.
AccountId	**integer.** The ID of the account whose orders will be returned

Response

The response returns an array of 1 or more order objects.

```
[
  {
    {
      "Side": {
         "Options": [
          "Buy",
          "Sell",
          "Short",
          "Unknown"
        ]
      },
      "OrderId": 0,
      "Price": 0,
      "Quantity": 0,
      "DisplayQuantity": 0,
      "Instrument": 0,
      "Account": 0,
      "OrderType": {
         "Options": [
          "Unknown",
          "Market",
          "Limit",
          "StopMarket",
          "StopLimit",
          "TrailingStopMarket",
          "TrailingStopLimit",
          "BlockTrade"
        ]
      },
      "ClientOrderId": 0,
      "OrderState": {
         "Options": [
          "Unknown",
```

Code continued on page 88

Code continued from page 87

```
                "Working",
                "Rejected",
                "Canceled",
                "Expired",
                "FullyExecuted"
              ]
            },
            "ReceiveTime": 0,
            "ReceiveTimeTicks": 0,
            "OrigQuantity": 0,
            "QuantityExecuted": 0,
            "AvgPrice": 0,
            "CounterPartyId": 0,
            "ChangeReason": {
               "Options": [
                "Unknown",
                "NewInputAccepted",
                "NewInputRejected",
                "OtherRejected",
                "Expired",
                "Trade",
                "SystemCanceled_NoMoreMarket",
                "SystemCanceled_BelowMinimum",
                "NoChange",
                "UserModified"
              ]
            },
            "OrigOrderId": 0,
            "OrigClOrdId": 0,
            "EnteredBy": 0,
            "IsQuote": false,
            "InsideAsk": 0,
            "InsideAskSize": 0,
            "InsideBid": 0,
            "InsideBidSize": 0,
            "LastTradePrice": 0,
            "RejectReason": "",
            "IsLockedIn": false,
            "OMSId": 0,
          },
        }
      ]
```

Where:

String	Value
Side	**string.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition).
OrderId	**long integer.** The ID of this order.
Price	**real.** Price of the order.
Quantity	**real.** Quantity of the order.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a DisplayQuantity value, an order must be a Limit order with a reserve. See "Display Quantity" on page 8, and "Order Types" on page 7.
Instrument	**integer.** The ID of the instrument being ordered.

Table continued on page 89

Table continued from page 88

Account	**integer.** The ID of the account ordering the instrument.
OrderType	**string.** One of: Unknown Market Limit StopMarket StopLimit TrailingStopMarket TrailingStopLimit BlockTrade See "Order Types" on page 7.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
OrderState	**string.** One of: Unknown Working Rejected Canceled Expired FullyExecuted An open order will probably not yet be fully executed.
ReceiveTime	**long integer.** The time at which the system received the quote, in POSIX format. See "Time– and Date-Stamp Formats" on page 8.
ReceiveTimeTicks	**long integer.** The time stamp of the received quote in Microsoft Ticks format. See "Time– and Date-Stamp Formats" on page 8.
OrigQuantity	**real.** If the order has been changed, this value shows the original quantity.
QuantityExecuted	**real.** This value states the quantity that was executed in the order. It may be the same as the quantity of the order; it may be different.
AvgPrice	**real.** Not currently used.
CounterPartyId	**long integer.** Shows 0.
ChangeReason	**string.** If the order has been changed, this value shows the reason. One of: Unknown NewInputAccepted NewInputRejected OtherRejected Expired Trade SystemCanceled_NoMoreMarket SystemCanceled_BelowMinimum NoChange UserModified
OrigOrderId	**integer.** If the order has been changed, shows the original order ID.
OrigClOrdId	**long integer.** If the order has been changed, shows the original client order ID, a value that the client can create (much like a purchase order).
EnteredBy	**integer.** The ID of the user who entered the order in this account.

Table continued on page 90

Table continued from page 89

IsQuote	**Boolean.** If this order is a quote (rather than an order), returns *true*, otherwise false. Default is *false*.
InsideAsk	**real.** Best Ask price available at time of entry (generally available to market makers).
InsideAskSize	**real.** Quantity available at the best inside ask price (generally available to market makers).
InsideBid	**real.** Best Bid price available at time of entry (generally available to market makers).
InsideBidSize	**real.** Quantity available at the best inside Bid price (generally available to market makers).
LastTradePrice	**real.** The price at which the instrument last traded.
RejectReason	**string.** If the order was rejected, this string value holds the reason.
IsLockedIn	**Boolean.** *True* if both parties to a block trade agree that one party will report the trade for both. Otherwise *false*.
OMSId	**integer.** The ID of the Order Management System on which the order was created.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOpenQuotes, GetOrderHistoryByOrderId, GetOrdersHistory, GetOrderStatus, ModifyOrder, SendOrder

GetOrderHistoryByOrderId

Retrieves the full order history of a specific order by its order ID, including any changes.

Request

```
{
 “OMSId”: 0,
 “OrderId”: 0,
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the orders were placed.
OrderId	**integer.** The ID of the order on the Order Management System.

Response

The response returns an array of 1 or more order objects.

```
[
  {
    {
      “Side”: {
         “Options”: [
          “Buy”,
          “Sell”,
          “Short”,
          “Unknown”
        ]
      },
      “OrderId”: 0,
      “Price”: 0,
      “Quantity”: 0,
      “DisplayQuantity”: 0,
      “Instrument”: 0,
      “Account”: 0,
      “OrderType”: {
         “Options”: [
          “Unknown”,
          “Market”,
          “Limit”,
          “StopMarket”,
          “StopLimit”,
          “TrailingStopMarket”,
          “TrailingStopLimit”,
          “BlockTrade”
        ]
      },
      “ClientOrderId”: 0,
      “OrderState”: {
         “Options”: [
          “Unknown”,
          “Working”,
```

Code continued on page 92

Code continued from page 91

```
            “Rejected”,
            “Canceled”,
            “Expired”,
            “FullyExecuted”
          ]
        },
        “ReceiveTime”: 0,
        “ReceiveTimeTicks”: 0,
        “OrigQuantity”: 0,
        “QuantityExecuted”: 0,
        “AvgPrice”: 0,
        “CounterPartyId”: 0,
        “ChangeReason”: {
           “Options”: [
            “Unknown”,
            “NewInputAccepted”,
            “NewInputRejected”,
            “OtherRejected”,
            “Expired”,
            “Trade”,
            “SystemCanceled_NoMoreMarket”,
            “SystemCanceled_BelowMinimum”,
            “NoChange”,
            “UserModified”
          ]
        },
        “OrigOrderId”: 0,
        “OrigClOrdId”: 0,
        “EnteredBy”: 0,
        “IsQuote”: false,
        “InsideAsk”: 0,
        “InsideAskSize”: 0,
        “InsideBid”: 0,
        “InsideBidSize”: 0,
        “LastTradePrice”: 0,
        “RejectReason”: “”,
        “IsLockedIn”: false,
        “OMSId”: 0,
      },
    }
  ]
```

Where:

String	Value
Side	**string.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
OrderId	**long integer.** The ID of this order.
Price	**real.** Price of the order.
Quantity	**real.** Quantity of the order.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a *DisplayQuantity* value, an order must be a Limit order with a reserve. See “Display Quantity” on page 8, and “Order Types” on page 7.
Instrument	**integer.** The ID of the instrument being ordered.

Table continued on page 93

Table continued from page 92

Account	**integer.** The ID of the account ordering the instrument.
OrderType	**string.** One of: Unknown Market Limit StopMarket StopLimit TrailingStopMarket TrailingStopLimit BlockTrade See “Order Types” on page 7.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
OrderState	**string.** One of: Unknown Working Rejected Canceled Expired FullyExecuted An open order will probably not yet be fully executed.
ReceiveTime	**long integer.** The time at which the system received the quote, in POSIX format. See “Time– and Date-Stamp Formats” on page 8.
ReceiveTimeTicks	**long integer.** The time stamp of the received quote in Microsoft Ticks format. See “Time– and Date-Stamp Formats” on page 8.
OrigQuantity	**real.** If the order has been changed, this value shows the original quantity.
QuantityExecuted	**real.** This value states the quantity that was executed in the order. It may be the same as the quantity of the order; it may be different.
AvgPrice	**real.** Not currently used.
CounterPartyId	**long integer.** Shows 0.
ChangeReason	**string.** If the order has been changed, this value shows the reason. One of: Unknown NewInputAccepted NewInputRejected OtherRejected Expired Trade SystemCanceled_NoMoreMarket SystemCanceled_BelowMinimum NoChange UserModified
OrigOrderId	**integer.** If the order has been changed, shows the original order ID.
OrigClOrdId	**long integer.** If the order has been changed, shows the original client order ID, a value that the client can create (much like a purchase order).
EnteredBy	**integer.** The ID of the user who entered the order in this account.

Table continued on page 94

Table continued from page 93

IsQuote	**Boolean.** If this order is a quote (rather than an order), returns *true*, otherwise *false*. Default is *false*.
InsideAsk	**real.** Best Ask price available at time of entry (generally available to market makers).
InsideAskSize	**real.** Quantity available at the best inside ask price (generally available to market makers).
InsideBid	**real.** Best Bid price available at time of entry (generally available to market makers).
IndisdeBidSize	**real.** Quantity available at the best inside Bid price (generally available to market makers)..
LastTradePrice	**real.** The price at which the instrument last traded.
RejectReason	**string.** If the order was rejected, this string value holds the reason.
IsLockedIn	**Boolean.** True if both parties to a block trade agree that one party will report the trade for both. Otherwise false.
OMSId	**integer.** The ID of the Order Management System on which the order was created.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOpenQuotes, GetOrderHistory, GetOrdersHistory, GetOrderStatus, ModifyOrder, SendOrder

GetOrdersHistory

Retrieves a history of multiple orders (hence, **GetOrdersHistory** with plural Orders) for the specified account, order ID, user, instrument, or time stamp, starting at index *i*, where *i* is an integer identifying a specific order in reverse order; that is, the most recent order has an index of 0. "Depth" is the count of trades to report backwards from *StartIndex*. All values in the call other than OMSId are optional.

The owner of the trading venue determines how long to retain order history before archiving.

Note: In this call, "Depth" refers not to the depth of the order book, but to the count of trades to report.

Request

All values other than OMSId are optional.

```
{
  "OMSId": 0,
  "AccountId": 0,
  "ClientOrderId": 0,
  "OriginalOrderId": 0,
  "OriginalClientOrderId": 0,
  "UserId": 0,
  "InstrumentId": 0,
  "StartTimestamp": 0,
  "EndTimestamp": 0,
  "Depth": 0,
  "StartIndex": 0,
}
```

Where:

String	Value
OMSId	**Integer.** The ID of the Order Management System on which the orders took place. Required. If no other values are specified, returns the orders associated with the default account for the logged-in user on this Order Management System.
AccountId	**Integer.** The account ID that made the trades. The logged-in user must be associated with this account, although other users also can be associated with the account. If no account ID is supplied, the system assumes the default account for the logged-in user.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
OriginalOrderId	**integer.** The original ID of the order. If specified, the call returns changed orders associated with this order ID.
UserId	**integer.** The ID of the user whose account orders will be returned. If not specified, the call returns the orders of the logged-in user.
InstrumentId	**long integer.** The ID of the instrument named in the order. If not specified, the call returns orders for all instruments for this account.

Table continued on page 96

Table continued from page 95

StartTimestamp	**long integer.** Date and time at which to begin the orders history, in POSIX format, and UTC time zone. If not specified, reverts to the start date of this account on the trading venue. See "Time– and Date-Stamp Formats" on page 8.
EndTimestamp	**long integer.** Date and time at which to end the orders report, in POSIX format, and UTC time zone. If not specified, uses the current time. See "Time– and Date-Stamp Formats" on page 8.
Depth	**integer.** In this case, the count of orders to return, counting from the *StartIndex*. If not specified, returns all orders between *BeginTimeStamp* and *EndTimeStamp*, beginning at *StartIndex* and working backwards.
StartIndex	**integer.** The starting index into the order history, from 0 (the most recent trade) and moving backwards in time. If not specified, defaults to 0.

Response

The response returns an array of order objects.

```
[
    {
      "Side": {
         "Options": [
          "Buy",
          "Sell",
          "Short",
          "Unknown"
        ]
      },
      "OrderId": 0,
      "Price": 0,
      "Quantity": 0,
      "DisplayQuantity": 0,
      "Instrument": 0,
      "Account": 0,
      "OrderType": {
         "Options": [
          "Unknown",
          "Market",
          "Limit",
          "StopMarket",
          "StopLimit",
          "TrailingStopMarket",
          "TrailingStopLimit",
          "BlockTrade"
        ]
      },
      "ClientOrderId": 0,
      "OrderState": {
         "Options": [
          "Unknown",
          "Working",
          "Rejected",
          "Canceled",
          "Expired",
          "FullyExecuted"
        ]
      },
      "ReceiveTime": 0,
      "ReceiveTimeTicks": 0,
      "OrigQuantity": 0,
      "QuantityExecuted": 0,
      "AvgPrice": 0,
      "CounterPartyId": 0,
      "ChangeReason": {
```

Code continued on page 97

Code continued from page 96

```
                    "Options":  [
                     "Unknown",
                     "NewInputAccepted",
                     "NewInputRejected",
                     "OtherRejected",
                     "Expired",
                     "Trade",
                     "SystemCanceled_NoMoreMarket",
                     "SystemCanceled_BelowMinimum",
                     "NoChange",
                     "UserModified"
                   ]
                 },
                 "OrigOrderId": 0,
                 "OrigClOrdId": 0,
                 "EnteredBy": 0,
                 "IsQuote": false,
                 "InsideAsk": 0,
                 "InsideAskSize": 0,
                 "InsideBid": 0,
                 "InsideBidSize": 0,
                 "LastTradePrice": 0,
                 "RejectReason": "",
                 "IsLockedIn": false,
                 "OMSId": 0,
               },
             }
           ]
```

Where:

String	Value
Side	**string.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
OrderId	**long integer.** The ID of this order.
Price	**real.** The unit price of the order.
Quantity	**real.** The quantity of the order.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a *DisplayQuantity* value, an order must be a Limit order with a reserve. See "Display Quantity" on page 8, and "Order Types" on page 7.
Instrument	**integer.** The ID of the instrument being ordered.
Account	**integer.** The ID of the account ordering the instrument.
OrderType	**string.** One of: Unknown Market Limit StopMarket StopLimit TrailingStopMarket TrailingStopLimit BlockTrade See "Order Types" on page 7.

Table continued on page 98

Table continued from page 97

ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
OrderState	**string.** One of: Unknown Working Rejected Canceled Expired FullyExecuted An open order will not be fully executed.
ReceiveTime	**long integer.** The time and date that the order was received, in POSIX format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
ReceiveTimeTicks	**long integer.** Identifies the time and date that the order was received in Microsoft ticks format, and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
OrigQuantity	**real.** The original quantity in the order (may be different from the amount executed).
QuantityExecuted	**real.** This value states the quantity that was executed in the order (may be different from *Quantity* or *OrigQuantity*.
AvgPrice	**real.** Not currently used.
CounterPartyId	**long integer.** Shows 0.
ChangeReason	**string.** If the order has been changed, this value shows the reason. One of: Unknown NewInputAccepted NewInputRejected OtherRejected Expired Trade SystemCanceled_NoMoreMarket SystemCanceled_BelowMinimum NoChange UserModified
OrigOrderId	**integer.** If the order has been changed, shows the original order ID.
OrigClOrdId	**long integer.** If the order has been changed, shows the original client order ID, a value that the client can create (much like a purchase order).
EnteredBy	**integer.** The ID of the user who entered the order in this account.
IsQuote	**Boolean.** If this order is a quote (rather than an order), returns *true*, otherwise *false*. Default is *false*.
InsideAsk	**real.** Best Ask price available at time of entry (generally available to market makers).
InsideAskSize	**real.** Quantity available at the best inside ask price (generally available to market makers).
InsideBid	**real.** Best Bid price available at time of entry (generally available to market makers).

Table continued on page 99

Table continued from page 98

IndisdeBidSize	**real.** Quantity available at the best inside Bid price (generally available to market makers)..
LastTradePrice	**real.** The price at which the instrument last traded.
RejectReason	**string.** If the order was rejected, this string value holds the reason.
IsLockedIn	**Boolean.** *True* if both parties to a block trade agree that one party will report the trade for both. Otherwise *false*.
OMSId	**integer.** The ID of the Order Management System on which the order was created.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOpenQuotes, GetOrderHistory, GetOrderHistoryByOrderId, GetOrderStatus, ModifyOrder, SendOrder

GetOrderStatus

Retrieves the status information for a single order.

Request

```
{
  "OMSId": 0,
  "AccountId": 0,
  "OrderId": 0,
 }
```

Where:

String	Value
OMSId	Integer. The ID of the Order Management System on which the order was placed.
AccountId	integer. The ID of the account under which the order was placed.
OrderId	integer. The ID of the order whose status will be returned.

Response

The response returns a single order object.

```
{
  "Side": {
     "Options": [
      "Buy",
      "Sell",
      "Short",
      "Unknown"
    ]
  },
  "OrderId": 0,
  "Price": 0,
  "Quantity": 0,
  "DisplayQuantity": 0,
  "Instrument": 0,
  "Account": 0,
  "OrderType": {
     "Options": [
      "Unknown",
      "Market",
      "Limit",
      "StopMarket",
      "StopLimit",
      "TrailingStopMarket",
      "TrailingStopLimit",
      "BlockTrade"
    ]
  },
  "ClientOrderId": 0,
  "OrderState": {
     "Options": [
      "Unknown",
```

Code continued on page 102

Code continued from page 101

```
                "Working",
                "Rejected",
                "Canceled",
                "Expired",
                "FullyExecuted"
              ]
            },
            "ReceiveTime": 0,
            "ReceiveTimeTicks": 0,
            "OrigQuantity": 0,
            "QuantityExecuted": 0,
            "AvgPrice": 0,
            "CounterPartyId": 0,
            "ChangeReason": {
               "Options": [
                "Unknown",
                "NewInputAccepted",
                "NewInputRejected",
                "OtherRejected",
                "Expired",
                "Trade",
                "SystemCanceled_NoMoreMarket",
                "SystemCanceled_BelowMinimum",
                "NoChange",
                "UserModified"
              ]
            },
            "OrigOrderId": 0,
            "OrigClOrdId": 0,
            "EnteredBy": 0,
            "IsQuote": false,
            "InsideAsk": 0,
            "InsideAskSize": 0,
            "InsideBid": 0,
            "InsideBidSize": 0,
            "LastTradePrice": 0,
            "RejectReason": "",
            "IsLockedIn": false,
            "OMSId": 0,
          }
```

Where:

String	Value
Side	**string.** The side of this order. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
OrderId	**long integer.** The ID of the order. The response echoes the order ID from the request.
Price	**real.** The price at which the order was placed.
Quantity	**real.** The quantity of the instrument being ordered.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a *DisplayQuantity* value, an order must be a Limit order with a reserve. See "Display Quantity" on page 8, and "Order Types" on page 7.
Instrument	**integer.** The ID of the instrument traded in the order.
Account	**integer.** The ID of the account that placed the order.

Table continued on page 103

Table continued from page 102

OrderType	**string.** One of: Unknown Market Limit StopMarket StopLimit TrailingStopMarket TrailingStopLimit BlockTrade See “Order Types” on page 7.
ClientOrderID	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
OrderState	**string.** One of: 0 Unknown 2 Working 3 Rejected 4 Canceled 5 Expired 6 FullyExecuted
ReceiveTime	**long integer.** The time and date that the order was received, in POSIX format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
ReceiveTimeTicks	**long integer.** Identifies the time and date that the order was received in Microsoft ticks format, and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
OrigQuantity	**real.** The original quantity of the order. The actual amount traded may be different.
QuantityExecuted	**real.** The quantity executed in this order. May be different from the amount ordered (*Quantity*).
AvgPrice	**real.** Not currently used.
CounterPartyId	**long integer.** Shows 0.
ChangeReason	**string.** The reason that the order may have been changed from the original. One of: 0 Unknown 1 NewInputAccepted 2 NewInputRejected 3 OtherRejected 4 Expired 5 Trade 6 SystemCanceled_NoMoreMarket 7 SystemCanceled_BelowMinimum 8 NoChange 9 UserModified
OrigOrderID	**integer.** The ID of the original order, if it has been changed.
OrigClOrId	**long inte**ger. If the order has been changed, shows the original client order ID, a value that the client can create (much like a purchase order). The default value is 0.
EnteredBy	**integer.** The ID of the user who originally entered the order.

Table continued on page 104

Table continued from page 103

IsQuote	**Boolean.** Returns *true* if the order is a quote, else returns *false*. Default is *false*. See “Quotes and Orders” on page 5.
InsideAsk	**real.** Ask price among market makers.
InsideAskSize	**real.** Ask quantity among market makers.
InsideBid	**real.** Bid price among market makers.
InsideBidSize	**real.** Bid quantity among market makers.
LastTradePrice	**real.** The price at which the instrument traded immediately before this trade.
RejectReason	**string.** If the trade was rejected, this string holds the reason.
IsLockedIn	**Boolean.** *True* if both parties to a block trade agree that one party will report the trade for both. Otherwise *false*.
OMSId	**integer.** ID of the Order Management System on which the trades being reported on occurred.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOpenQuotes, GetOrderHistory, GetOrderHistoryByOrderId, GetOrdersHistory, ModifyOrder, SendOrder

GetProduct

No authentication required

Retrieves the details about a specific product on the trading venue. A *product* is an asset that is tradable or paid out. See “Products and Instruments” on page 4 for more information about the difference between these two items.

Request

```
{
  “OMSId”:  1,
  “ProductId”:  1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System that includes the product.
ProductId	**long integer.** The ID of the product (often a currency) on the specified Order Management System.

Response

Unsuccessful response:

```
{
  “OMSId”: 0,
  “ProductId”: 0,
  “Product”: “”,
  “ProductFullName”: “”,
  “ProductType”: {
    “Options”:  [
      “Unknown”,
      “NationalCurrency”,
      “CryptoCurrency”,
      “Contract”
    ]
  },
  “DecimalPlaces”: 0,
  “TickSize”: 0,
  “NoFees”: false,

}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System that offers the product.
ProductId	**long integer.** The ID of the product.

Product	**string.** “Nickname” or shortened name of the product. For example, NZD (New Zealand Dollar).
ProductFullName	**string.** Full and official name of the product. For example, New Zealand Dollar.
ProductType	**string.** The nature of the product. One of: 0 Unknown (an error condition) 1 NationalCurrency 2 CryptoCurrency 3 Contract
DecimalPlaces	**integer.** The number of decimal places in which the product is divided. For example, US Dollars are divided into 100 units, or 2 decimal places. Other products may be different. Burundi Francs use 0 decimal places and the Rial Omani uses 3.
TickSize	**integer.** Minimum tradable quantity of the product. See also **GetInstrument**, where this value is called *QuantityIncrement*. For example, with a US Dollar, the minimal tradable quantity is $0.01.
NoFees	**Boolean.** Shows whether trading the product incurs fees. The default is *false*; that is, if *NoFees* is *false*, fees will be incurred. If *NoFees* is *true*, no fees are incurred.

See Also

GetAccountPositions, GetInstrument, GetInstruments, GetProducts

GetProducts

No authentication required

Returns an array of products available on the trading venue. A *product* is an asset that is tradable or paid out. For more information about the difference between products and instruments, see “Products and Instruments” on page 4.

Request

```
{
 “OMSId”: 1,
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System for which the array of available products and currencies will be returned.

Response

The response returns an array of objects, one object for each product available on the Order Management System.

```
[
  {
  “OMSId”: 0,
  “ProductId”: 0,
  “Product”: “”,
  “ProductFullName”: “”,
  “ProductType”: {
    “Options”:  [
        “Unknown”,
        “NationalCurrency”,
        “CryptoCurrency”,
        “Contract”
      ]
    },
  “DecimalPlaces”: 0,
  “TickSize”: 0,
  “NoFees”: false,
  },
]
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System that offers the product.
ProductId	**long integer.** The ID of the product.

Product	**string.** “Nickname” or shortened name of the product. For example, NZD (New Zealand Dollar).
ProductFullName	**string.** Full and official name of the product. For example, New Zealand Dollar.
ProductType	**string.** The nature of the product. One of: 0 Unknown (an error condition) 1 NationalCurrency 2 CryptoCurrency 3 Contract
DecimalPlaces	**integer.** The number of decimal places in which the product is divided. For example, US Dollars are divided into 100 units, or 2 decimal places. Other products may be different. Burundi Francs use 0 decimal places and the Rial Omani uses 3.
TickSize	**integer.** Minimum tradable quantity of the product. See also **GetInstrument**, where this value is called *QuantityIncrement*. For example, with a US Dollar, the minimal tradable quantity is $0.01.
NoFees	**Boolean.** Shows whether trading the product incurs fees. The default is *false*; that is, if *NoFees* is *false*, fees will be incurred. If *NoFees* is *true*, no fees are incurred.

See Also

GetAccountPositions, GetInstrument, GetInstruments, GetProduct

ModifyOrder

Reduces an order's quantity without losing priority in the order book. *An order's quantity can only be reduced.* The other call that can modify an order — **CancelReplaceOrder** — resets order book priority, but you can use it to increase an order.

Note: **ModifyOrder** does not surrender or reset order book priority.

Request

```
{
   "OMSId": 0,
   "OrderId": 0,
   "InstrumentId": 0,
   "PreviousOrderRevision": 0,
   "Quantity": 0
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the original order was placed.
OrderId	**long integer.** The ID of the order to be modified. The ID was supplied by the server when the order was created.
InstrumentId	**integer.** The ID of the instrument traded in the order.
PreviousOrderRevision	**integer.** The order revision number at the time you make the modification order. This ensures that you have the latest order state at the time you make the request.
Quantity	**real.** The new quantity of the order. This value can only be reduced from a previous quantity.

Response

```
{
  "result": false,
  "errormsg": "",
  "errorcode": 0,
  "detail": "",
}
```

Where:

String	Value
result	**Boolean.** The successful receipt of a modify request returns *true*; otherwise, returns *false*. This is the acknowledgement of receipt of the request to modify, not a confirmation that the modification has taken place. Monitor the modification with **GetOpenOrders** or **GetOrderHistory**.
errormsg	**string.** A successful receipt of a modify request returns *null*; the *errormsg* parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** The receipt of a successful request to modify returns 0. An unsuccessful request returns one of the errorcodes shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOrderHistory, GetOrderHistoryByOrderId, GetOrdersHistory, GetOrderStatus, SendOrder

SendOrder

Creates an order. Anyone submitting an order should also subscribe to the various market data and event feeds, or call **GetOpenOrders** or **GetOrderStatus** to monitor the status of the order. If the order is not in a state to be executed, **GetOpenOrders** will not return it.

Request

```
{
  "AccountId": 5,
  "ClientOrderId": 99,
  "Quantity": 1,
  "DisplayQuantity": 0,
  "UseDisplayQuantity": true,
  "LimitPrice": 95,
  "OrderIdOCO": 0,
  "OrderType": 2,
  "PegPriceType": 1,
  "InstrumentId": 1,
  "TrailingAmount": 1.0,
  "LimitOffset": 2.0,
  "Side": 0,
  "StopPrice": 96,
  "TimeInForce": 1,
  "OMSId": 1,
}
```

Where:

String	Value
AccountId	**integer.** The ID of the account placing the order.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). This ID is useful for recognizing future states related to this order. *ClientOrderId* defaults to 0.
Quantity	**real.** The quantity of the instrument being ordered.
DisplayQuantity	**real.** The quantity available to buy or sell that is publicly displayed to the market. To display a *DisplayQuantity* value, an order must be a Limit order with a reserve. See "Display Quantity" on page 8.
UseDisplayQuantity	**Boolean.** If you enter a Limit order with a reserve, you must set *UseDisplayQuantity* to *true*. See "Display Quantity" on page 8 for more information about how the system users the *DisplayQuantity* value.
LimitPrice	**real.** The price at which to execute the order, if the order is a Limit order.
OrderIdOCO	**integer.** *One Cancels the Other* — If this order is order A, *OrderIdOCO* refers to the order ID of an order B (which is not the order being created by this call). If order B executes, then order A created by this call is canceled. You can also set up order B to watch order A in the same way, but that may require an update to order B to make it watch this one, which could have implications for priority in the order book. See **CancelReplaceOrder** and **ModifyOrder**.

Table continued on page 118

Table continued from page 117

OrderType	**integer.** The type of this order, as expressed in integer format. See "Order Types" on page 7 for an explanation of each type. One of: 1 Market 2 Limit 3 StopMarket 4 StopLimit 5 TrailingStopMarket 6 TrailingStopLimit 7 BlockTrade.
PegPriceType	**integer.** When entering a stop/trailing order, set *PegPriceType* to an integer that corresponds to the type of price that pegs the stop: 1 Last 2 Bid 3 Ask 4 Midpoint
InstrumentId	**long integer.** The ID of the instrument being traded in the order.
TrailingAmount	**real.** The offset by which to trail the market in one of the trailing order types. Set this to the current price of the market to ensure that the trailing offset is the amount intended in a fast-moving market. See "Order Types" on page 7.
LimitOffset	**real.** The amount by which a trailing limit order is offset from the activation price.
Side	**integer.** The side of the trade represented by this order. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
StopPrice	**real.** The price at which to execute the order, if the order is a Stop order (either buy or sell).
TimeInForce	**integer.** The period during which the order is executable. 0 Unknown (error condition) 1 GTC good 'til canceled 3 IOC immediate or cancelled 4 FOK fill or kill — fill the order immediately, or cancel it immediately There may be other settings for TimeInForce depending on the trading venue.
OMSId	**integer.** The ID of the Order Management System on which the order is being placed.

Response

```
{
  "status":"Accepted",
  "errormsg":"",
  "OrderId": 123 // Server order id
}
```

Where:

String	Value
status	**string.** If the order is accepted by the system, it returns 0. 0 Accepted 1 Rejected
errormsg	**string.** Any error message the server returns.
OrderId	**long integer.** The ID assigned to the order by the server. This allows you to track the order.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOrderHistory, GetOrderHistoryByOrderId, GetOrdersHistory, GetOrderStatus, ModifyOrder

UpdateQuote

Updates an existing quote. Quoting is not enabled for the retail end user of the Coinext software. Only registered market participants or market makers may quote. See **CancelReplaceOrder**.

Note: **UpdateQuote** resets the quote's priority in the order book.

Request

```
{
  "OMSId": 0,
  "AccountId": 0,
  "InstrumentId": 0,
  "BidQuoteId": 0,
  "Bid": 0,
  "BidQty": 0,
  "AskQuoteId": 0,
  "Ask": 0,
  "AskQty": 0,
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the quote is located.
AccountId	**integer.** The ID of the account whose quote will be updated.
InstrumentId	**long integer.** The ID of the instrument whose quote is being updated.
BidQuoteId	**integer.** The ID of the original bid quote being updated.
Bid	**real.** The new currency amount of the bid quote.
BidQty	**real.** The new quantity of the bid quote.
AskQuoteId	**integer.** The ID of the original ask quote being updated.
Ask	**real.** The new currency amount of the ask quote.
AskQty	**real.** The new quantity of the ask quote.

Response

```
{
  "BidQuoteId": 0,
  "BidResult": "{
    "result": true,
    "errormsg": "",
    "errorcode": 0,
                "detail": "",
              }",
              "AskQuoteId": 0,
              "AskResult": "{
                "result": true,
                "errormsg": "",
                "errorcode": 0,
                "detail": "",
              }"
            }
```

Where:

String	Value
BidQuoteId	integer. The ID of the Bid quote being updated.
BidResult	string. Returns a response object for Bid.
AdkQuoteId	integer. The ID of the Ask quote being updated.
AskResult	string. Returns a response object for Ask.

Response objects for both *BidResult* and *AskResult*:

String	Value
result	**Boolean.** A successful receipt of the update returns *true*; and unsuccessful receipt of the update (an error condition) returns *false*.
errormsg	**string.** A successful receipt of the update returns *null*; the *errormsg* string for an unsuccessful receipt returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the update returns 0. An unsuccessful receipt returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

CancelAllOrders, CancelOrder, CancelQuote, CancelReplaceOrder, CreateQuote, GetOpenOrders, GetOpenQuotes, ModifyOrder, SendOrder

Reports

CancelUserReport

You can generate or schedule a variety of reports through this API on demand. This call cancels a scheduled report by its report ID.

Request

GetUserReportTickets can provide a list of GUIDs for scheduled reports.

```
{
 "UserReportId": guid-as-a-string //GUID not GUIDE
}
```

Where:

String	Value
UserReport	**string.** The GUID is a globally unique ID string that identifies the user report to be cancelled. The Order Management System provides this ID when you create a report.

Response

The response to **CancelUserReport** verifies that the call was received, not that the user report has been canceled successfully. Individual event updates to the user show cancellations. To verify that a report has been canceled, call **GetUserReportTickets** or **GetUserReportWriterResultRecords**.

```
{
  "result": true,
  "errormsg": "",
  "errorcode": 0,
  "detail": "",
}
```

Where:

String	Value
result	**Boolean.** A successful receipt of the cancellation returns true; and unsuccessful receipt of the cancellation (an error condition) returns false.
errormsg	**string.** A successful receipt of the cancellation returns null; the errormsg parameter for an unsuccessful receipt returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the cancellation returns 0. An unsuccessful receipt returns one of the errorcodes shown in the *errormsg* list.

Table continued on page 126

Table continued from page 125

detail	**string.** Message text that the system may send. Usually *null*.

See Also

GetUserReportTickets, GetUserReportWriterResultRecords, ScheduleTradeActivityReport, ScheduleTransactionActivityReport, ScheduleTreasuryActivityReport

GenerateTradeActivityReport

Creates an immediate report on historical trade activity on a specific Order Management System for a list of accounts during a specified time interval.

The accounts listed in the request must all be associated with the logged-in user on the specified OMS (the logged-in user may not be the only user of each account).

The Trade Activity Report is delivered as a comma-separated (CSV) file. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  "accountIdList": [
    0 // one or more Account IDs
  ],
  "omsId": 0,
  "startTime": "0001-01-01T05:00:00Z".
  "endTime": "0001-01-01T05:00:00Z",
}
```

Where:

String	Value
accountIdList	**integer array**. A comma-delimited array of one ore more account IDs, each valid on a single Order Management System for the authenticated user. The account user may not be the only user of the account. See "Permissions" on page 4.
omsId	**integer.** The ID of the Order Management System on which the array of account IDs exist.
startTime	**string.** *startTime* identifies the time and date for the historic beginning of the trade activity report in ISO 8601 format and UTC time zone. "Time– and Date-Stamp Formats" on page 8.
endTime	**string.** *endTime* identifies the time and date for the historic end of the trade activity report in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.

Response

Similar objects are returned for **Generate~Report** and **Schedule~Report** calls. As a result, for an on-demand **Generate~Report** call, some string-value pairs such as *initialRunTime* may return the current time and *ReportFrequency* will always return *OnDemand* because the report is only generated once and on demand.

```
{
  "RequestingUser": 0,
  "OMSId": 0,
  "reportFlavor": {
     "Options": [
      "TradeActivity",
      "Transaction",
      "Treasury"
```

Code continued on page 128

Code continued from page 127

```
      ]
    },
    "createTime": "0001-01-01T05:00:00Z",
    "initialRunTime": "0001-01-01T05:00:00Z",
    "intervalStartTime": "0001-01-01T05:00:00Z",
    "intervalEndTime": "0001-01-01T05:00:00Z",
    "RequestStatus": {
       "Options": [
        "Submitted",
        "Validating",
        "Scheduled",
        "InProgress",
        "Completed",
        "Aborting",
        "Aborted",
        "UserCancelled",
        "SysRetired",
        "UserCancelledPending"
      ]
    },
    "ReportFrequency": {
       "Options":  [
        "onDemand",
        "Hourly",
        "Daily",
        "Weekly",
        "Monthly",
        "Annually"
      ]
    },
    "intervalDuration": 0,
    "RequestId": "AAAAAAAAAAAAAAAAAAAAAA==",
    "lastInstanceId": "AAAAAAAAAAAAAAAAAAAAAA==",
    "accountIds": [
      0
    ],
  }
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the trade activity report. This confirms the ID of the authenticated user who made the request by returning it as part of the response.
OMSId	**integer.** The ID of the Order Management System on which the trade activity report will be run.
reportFlavor	**string.** The type of report to be generated. One of: TradeActivity Transaction Treasury The *reportFlavor* string confirms the nature of the call.
createTime	**string.** The time and date on which the request for the trade activity report was made, in ISO 8601 format and the UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
initialRunTime	**string.** The time and date at which the trade activity report was first run, in ISO 8601 format and the UTC time zone. Returns the current time for a Generate~Report call. See "Time– and Date-Stamp Formats" on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format. See "Time– and Date-Stamp Formats" on page 8.

Table continued on page 128

Table contnued from page 128

intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format. See "Time– and Date-Stamp Formats" on page 8.
requestStatus	**string.** The status of the request for the trade activity report. A **Generate~Report** request will always return *Submitted*. See "Request Status" on page 10. Each request returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPending
ReportFrequency	**string.** When the report runs. For a **Generate~Report** call, this is always *OnDemand*. OnDemand Hourly Daily Weekly Monthly Annually
intervalDuration	**long integer.** The period that the report covers relative to the run date, expressed in Microsoft ticks format. The **Generate~Report** call requires a start time and an end time. The Coinext software calculates the difference between them as *intervalDuration*. See ""Time– and Date-Stamp Formats" on page 8. For example, say that you specify a 90-day start-date-to-end-date window for a report. The *intervalDuration* value returns a value equivalent to 90 days. If you have called **Generate~Report**, that value simply confirms the length of time that the on-demand report covers.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management Systsem.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report. Will be null for a **Generate~Report** call, because generated reports are on-demand.
accountId	**integer array.** A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTransactionActivityReport, GenerateTreasuryActivityReport, GetUserReportTickets, GetUserReportWriterResultRecords, ScheduleTradeActivityReport, ScheduleTransactionActivityReport, ScheduleTreasuryActivityReport

GenerateTransactionActivityReport

Generates an immediate report on account transaction activity for a list of accounts under a single Order Management System during a specified time. A logged-in and authenticated user can only generate a transaction activity report for accounts associated with the user. There can be multiple users associated with an account however; see “Permissions” on page 4.

The Transaction Activity Report is delivered as a comma-separated (CSV) file. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  “accountIdList”: [
    0
  ],
  “omsId”: 0,
  “startTime”: “0001-01-01T05:00:00Z”,
  “endTime”: “0001-01-01T05:00:00Z”,
}
```

Where:

String	Value
accountIdList	**integer array.** A comma-deliminted array of one ore more account IDs, each valid on the same Order Management System on which the user is authenticated.
omsId	**integer.** The ID of the Order Management System on which the array of account IDs exist.
startTime	**string.** *startTime* identifies the time and date for the beginning of the transaction activity report, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.
endTime	**string.** *endTime* identifies the time and date for the end of the transaction activity report, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.

Response

Similar objects are returned for **Generate~Report** and **Schedule~Report** calls. As a result, for an on-demand **Generate~Report** call, some string-value pairs such as *initialRunTime* may return the current time and *ReportFrequency* will always return *OnDemand* because the report is only generated once and on demand.

```
{
  “RequestingUser”: 0,
  “OMSId”: 0,
  “reportFlavor”: {
     “Options”:  [
      “TradeActivity”,
      “Transaction”,
      “Treasury”
    ]
  },
```

Code continued on page 132

Code continued from page 131

```
        “createTime”: “0001-01-01T05:00:00Z”,
        “initialRunTime”: “0001-01-01T05:00:00Z”,
        “intervalStartTime”: “0001-01-01T05:00:00Z”,
        “intervalEndTime”: “0001-01-01T05:00:00Z”,
        “RequestStatus”: {
           “Options”: [
            “Submitted”,
            “Validating”,
            “Scheduled”,
            “InProgress”,
            “Completed”,
            “Aborting”,
            “Aborted”,
            “UserCancelled”,
            “SysRetired”,
            “UserCancelledPending”
          ]
        },
        “ReportFrequency”: {
           “Options”:  [
            “onDemand”,
            “Hourly”,
            “Daily”,
            “Weekly”,
            “Monthly”,
            “Annually”
          ]
        },
        “intervalDuration”: 0,
        “RequestId”: “AAAAAAAAAAAAAAAAAAAAAA==”,
        “lastInstanceId”: “AAAAAAAAAAAAAAAAAAAAAA==”,
        “accountIds”: [
          0
        ],
      }
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the transaction activity report. This confirms the ID of the authenticated user who made the request by returning it as part of the response..
OMSId	**integer.** The ID of the Order Management System on which the transaction activity report will be run.
reportFlavor	**string.** The type of report to be generated. One of: TradeActivity Transaction Treasury The *reportFlavor* string confirms the nature of the call.
createTime	**string.** The time and date on which the request for the trade activity report was made, in ISO 8601 format and the UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
initialRunTime	**string.** The time and date at which the trade activity report was first run, in ISO 8601 format and the UTC time zone. Returns the current time for a **Generate~Report** call. See “Time– and Date-Stamp Formats” on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.

Table continued on page 132

Table continued from page 132

intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format. See "Time– and Date-Stamp Formats" on page 8.
requestStatus	**string.** The status of the request for the trade activity report. A **Generate~Report** request will always return *Submitted*. See "Request Status" on page 10. Each request returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPending
ReportFrequency	**string.** When the report runs. For a **Generate~Report** call, this is always *OnDemand*. OnDemand Hourly Daily Weekly Monthly Annually
intervalDuration	**long integer.** The period that the report covers relative to the run date, expressed in Microsoft ticks format. The **Generate~Report** call requires a start time and an end time. The Coinext software calculates the difference between them as *intervalDuration*. See "Time– and Date-Stamp Formats" on page 8. For example, say that you specify a 90-day start-date-to-end-date window for a report. The *intervalDuration* value returns a value equivalent to 90 days. If you have called **Generate~Report,** that value simply confirms the length of time that the on-demand report covers.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management Systsem.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report. Will be null for a **Generate~Report** call, because generated reports are on-demand.
accountIds	**integer array.** A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTradeActivityReport, GenerateTreasuryActivityReport, GetUserReportTickets, GetUserReportWriterResultRecords, ScheduleTradeActivityReport, ScheduleTransactionActivityReport, ScheduleTreasuryActivityReport

GenerateTreasuryActivityReport

Generates an immediate report on all company treasury activities related to the trading venue — withdrawals, transfers, and funds movements unrelated to trading — over a specified period. A logged-in and authenticated user can only generate a transaction activity report for accounts associated with the user. There can be multiple users associated with an account; see "Permissions" on page 4.

The Trade Activity Report is delivered as a comma-separated (CSV) file. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  "accountIdList": [
    0
  ],
  "omsId": 0,
  "startTime": "0001-01-01T05:00:00Z",
  "endTime": "0001-01-01T05:00:00Z",
}
```

Where:

String	Value
accountIdList	**integer array.** A comma-delimited array of one ore more account IDs, each valid on a single Order Management System for the authenticated user. The account user may not be the only user of the account. See "Permissions" on page 4.
omsId	**integer.** The ID of the Order Management System on which the array of account IDs exist.
startTime	**string.** *startTime* identifies the time and date for the historic beginning of the trade activity report in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
endTime	**string.** *endTime* identifies the time and date for the historic end of the trade activity report in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.

Response

Similar objects are returned for **Generate~Report** and **Schedule~Report** calls. As a result, for an on-demand **Generate~Report** call, some string-value pairs such as *initialRunTime* may return the current time and *ReportFrequency* will always return *OnDemand* because the report is only generated once and on demand.

```
{
  "OMSId": 0,
  "reportFlavor": {
     "Options":  [
      "TradeActivity",
      "Transaction",
      "Treasury"
```

Code continued on page 136

Code continued from page 135

```
    ]
  },
  "createTime": "0001-01-01T05:00:00Z",
  "initialRunTime": "0001-01-01T05:00:00Z",
  "intervalStartTime": "0001-01-01T05:00:00Z",
  "intervalEndTime": "0001-01-01T05:00:00Z",
  "RequestStatus": {
     "Options":  [
      "Submitted",
      "Validating",
      "Scheduled",
      "InProgress",
      "Completed",
      "Aborting",
      "Aborted",
      "UserCancelled",
      "SysRetired",
      "UserCancelledPending"
    ]
  },
  "ReportFrequency": {
     "Options":  [
      "onDemand",
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annually"
    ]
  },
  "intervalDuration": 0,
  "RequestId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "lastInstanceId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "accountIds": [
    0
  ],
}
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the treasury activity report. This confirms the ID of the authenticated user who made the request by returning it as part of the response.
OMSId	**integer.** The ID of the Order Management System on which the transaction activity report will be run.
reportFlavor	**string.** The type of report to be generated. One of: TradeActivity Transaction Treasury The reportFlavor string confirms the nature of the call.
createTime	**string.** The time and date on which the request for the trade activity report was made, in ISO 8601 format and the UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
initialRunTime	**string.** The time and date at which the trade activity report was first run, in ISO 8601 format and the UTC time zone. Returns the current time for a **Generate~Report** call. See "Time– and Date-Stamp Formats" on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format. See "Time– and Date-Stamp Formats" on page 8.

Table continued on page 137

Table continued from page 136

intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format. See "Time– and Date-Stamp Formats" on page 8.
requestStatus	**string.** The status of the request for the trade activity report. A Generate~Report request will always return Submitted. See "Request Status" on page 10. Each request returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPendin
ReportFrequency	**string.** When the report runs. For a **Generate~Report** call, this is always *OnDemand*. OnDemand Hourly Daily Weekly Monthly Annually
intervalDuration	**long integer.** The period that the report covers relative to the run date, expressed in Microsoft ticks format. The **Generate~Report** call requires a start time and an end time. The Coinext software calculates the difference between them as *intervalDuration*. See "Time– and Date-Stamp Formats" on page 8. For example, say that you specify a 90-day start-date-to-end-date window for a report. The *intervalDuration* value returns a value equivalent to 90 days. If you have called **Generate~Report**, that value simply confirms the length of time that the on-demand report covers.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management Systsem.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report. Will be null for a **Generate~Report** call, because generated reports are on-demand.
accountIds	**integer array.** A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTradeActivityReport, **GenerateTransactionActivityReport**, **GetUserReportTickets**, **ScheduleTradeActivityReport**, **GenerateTreasuryActivityReport**, *and* **ScheduleTreasuryActivityReport**.

GetUserReportTickets

Returns an array of user report tickets for a specific user ID. A user report ticket identifies a report requested or subscribed to by a user. Reports can run once or periodically.

Request

```
{
   “UserId”:  1
}
```

Where:

String	Value
UserId	**integer.** The ID of the user whose user report tickets will be returned.

Response

The response returns an array of tickets, each ticket representing a report.

```
[
  {
    {
      “RequestingUser”: 0,
      “OMSId”: 0,
      “reportFlavor”: {
         “Options”: [
          “TradeActivity”,
          “Transaction”,
          “Treasury”
      },
      “createTime”: “0001-01-01T05:00:00Z”,
      “initialRunTime”: “0001-01-01T05:00:00Z”,
      “intervalStartTime”: “0001-01-01T05:00:00Z”,
      “intervalEndTime”: “0001-01-01T05:00:00Z”,
      “RequestStatus”: {
         “Options”: [
          “Submitted”,
          “Validating”,
          “Scheduled”,
          “InProgress”,
          “Completed”,
          “Aborting”,
          “Aborted”,
          “UserCancelled”,
          “SysRetired”,
          “UserCancelledPending”
        ]
      },
      “ReportFrequency”: {
         “Options”: [
          “onDemand”,
          “Hourly”,
          “Daily”,
          “Weekly”,
          “Monthly”,
          “Annually”
```

Code continued on page 140

Code continued from page 139

```
                    ]
                  },
                  "intervalDuration": 0,
                  "RequestId": "AAAAAAAAAAAAAAAAAAAAAA==",
                  "lastInstanceId": "AAAAAAAAAAAAAAAAAAAAAA==",
                  "accountIds": [
                    0
                  ],
                },
              }
            ]
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the report.
OMSId	**integer.** The ID of the Order Management System on which the report was run.
reportFlavor	**string.** The type of report. One of: TradeActivity Transaction Treasury For more information, see "Report Types" on page 9.
createTime	**string.** The time and date on which the request for the report was made, in ISO 8601 format, and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
initialRunTime	**string.** The time and date at which the report was first run, in ISO 8601 format, and UTC time zone. See ""Time– and Date-Stamp Formats" on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format, and UTC time zone. See ""Time– and Date-Stamp Formats" on page 8.
intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format, and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
requestStatus	**string.** The status of the request for the report. See "Request Status" on page 10. Each status returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPending
ReportFrequency	**string.** When the report runs. OnDemand Hourly Daily Weekly Monthly Annually

Table continued on page 141

Table continued from page 140

intervalDuration	**long integer.** The period that the report looks backward relative to the run date. The system calculates *intervalDuration* between *intervalStartTime* and *intervalEndTime* and reports it in the form of Microsoft ticks. (See "Time– and Date-Stamp Formats" on page 8.) For example, say that you specify a 90-day start-date-to-end-date window for a report. The *intervalDuration* value returns a value equivalent to 90 days and represents the backward-looking period of the report. Say that you have set a weekly report to look back 90 days. When it runs again in a week's time, it again looks back 90 days — but now those 90 days are offset by a week from the first report.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management System.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report. Will be *null* for a **Generate~Report** call, because generated reports are on-demand.
accountIds	**integer array.** A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTradeActivityReport, **GenerateTransactionActivityReport**, **GenerateTreasuryActivityReport**, **GetUserReportWriterResultRecords, ScheduleTradeActivityReport**, **ScheduleTreasuryActivityReport**, **ScheduleTreasuryActivityReport**.

GetUserReportTickets

GetUserReportWriterResultRecords

The call returns an array of user report writer results. The results are the details of when reports have been run, and the status of each report run. Did the report complete? Did the report not start? The call requires no details. The call uses the default information from the logged-in and authenticated user.

Request

Requires no details.

```
{
  // no request details are needed
}
```

Response

```
[
  {
    {
      "RequestingUser": 0,
      "urtTicketId": "AAAAAAAAAAAAAAAAAAAAAA==",
      "descriptorId": "AAAAAAAAAAAAAAAAAAAAAA==",
      "resultStatus": {
         "Options": [
          "NotStarted",
          "NotComplete",
          "ErrorComplete",
          "SuccessComplete",
          "Cancelled"
        ]
      },
      "reportExecutionStartTime": "0001-01-01T05:00:00Z",
      "reportExecutionCompleteTime": "0001-01-01T05:00:00Z",
      "reportDescriptiveHeader": "",
    },
  }
]
```

Where:

String	Value
RequestingUser	**Integer.** ID of the logged-in user requesting the report.
urtTicketId	**string.** An alphanumeric string containing the unique report ID of the report.
descriptorId	**string.** A GUID (globally-unique identifier) that describes the report separately from the report ticket.
resultStatus	**string.** The status of each run of the reports. One of: 0 NotStarted 1 NotComplete 2 ErrorComplete 3 SuccessComplete 4 Cancelled

Table continued on page 144

Table continued from page 143

reportExecutionStartTime	**long integer.** The time that the report writer began execution, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
reportExecution-CompleteTime	**long integer.** The time that the report writer completed the report, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
reportDescriptiveHeader	**string.** A string describing the report.

See Also

GenerateTradeActivityReport, **GenerateTransactionActivityReport**, **GenerateTreasuryActivityReport**, **GetUserReportTickets, ScheduleTradeActivityReport**, **ScheduleTreasuryActivityReport**, **ScheduleTreasuryActivityReport**.

ScheduleTradeActivityReport

Schedules a series of trade activity reports to run for a list of accounts on a single Order Management System, starting at a specific date/time, and covering a specific time duration. The reports will run periodically until canceled.

Trade Activity Reports are delivered in comma-separated-value (CSV) format. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  "accountIdList": [
    0
  ],
  "omsId": 0,
  "beginTime": "0001-01-01T05:00:00Z",
  "intervalDuration": 0,
  "frequency": {
     "Options": [
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annual"
    ]
  },
}
```

Where:

String	Value
AccountIdList	**integer array.** Comma-separated integers; each element an account ID on the Order Management System whose trade activity will be reported on. All accounts must be from the same OMS and be associated with the logged-in user.
OMSId	**integer.** The Order Management System on which the accounts named in the list reside.
beginTime	**string.** The time from which the trade activities will be reported, in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
intervalDuration	**integer.** The length of time prior to the run time that the report covers, in Microsoft ticks format. For example, 90 days. Whenever the report runs, it looks back 90 days.
frequency	**string.** How often the report will run. One of: 0 OnDemand 1 Hourly 2 Daily 3 Weekly 4 Monthly 5 Annually

Response

The response returns an object confirming the settings in the call.

```
{
  "RequestingUser": 0,
  "OMSId": 0,
  "reportFlavor": {
     "Options": [
      "TradeActivity",
      "Transaction",
      "Treasury"
    ]
  },
  "createTime": "0001-01-01T05:00:00Z",
  "initialRunTime": "0001-01-01T05:00:00Z",
  "intervalStartTime": "0001-01-01T05:00:00Z",
  "intervalEndTime": "0001-01-01T05:00:00Z",
  "RequestStatus": {
     "Options": [
      "Submitted",
      "Validating",
      "Scheduled",
      "InProgress",
      "Completed",
      "Aborting",
      "Aborted",
      "UserCancelled",
      "SysRetired",
      "UserCancelledPending"
    ]
  },
  "ReportFrequency": {
     "Options": [
      "onDemand",
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annually"
    ]
  },
  "intervalDuration": 0,
  "RequestId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "lastInstanceId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "accountIds": [
    0
  ],
}
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the trade activity report. This confirms the ID of the authenticated user who made the request by returning it as part of the response.
OMSId	**integer.** The ID of the Order Management System on which the trade activity report will be run.
reportFlavor	**string.** The type of report to be generated. One of: TradeActivity Transaction Treasury The *reportFlavor* string confirms the nature of the call.

Table continued on page 147

Table continued from page 146

createTime	**string.** The time and date on which the request for the trade activity report was made, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
initialRunTime	**string.** The time and date at which the trade activity report was first run, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
requestStatus	**string.** The status of the request for the trade activity report. See “Request Status” on page 10. Each request returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPending
ReportFrequency	**string.** When the report runs. OnDemand Hourly Daily Weekly Monthly Annually
intervalDuration	**long integer.** The period that the report covers relative to the run date. The call specifies a start time and an *intervalDuration* in the form of Microsoft ticks. (See “Time– and Date-Stamp Formats” on page 8.) For example, say that you schedule a weekly report with a 90-day *intervalDuration* value. *intervalDuration* represents the backward-looking period of the report. When the report runs again in a week’s time, it again looks back 90 days — but now those 90 days are offset by a week from the first report.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management Systsem.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report.
accountIds	**integer array.** A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTradeActivityReport, **GenerateTransactionActivityReport**, **GenerateTreasuryActivityReport**, **GetUserReportTickets**, **GetUserReportWriterResultRecords**, **ScheduleTreasuryActivityReport**, **ScheduleTreasuryActivityReport**.

ScheduleTransactionActivityReport

Schedules a series of transaction activity reports for a list of accounts on a single Order Management System, starting at a specific date/time, and covering a specific time interval (90 days, for example). The reports will run periodically until canceled.

Transaction Activity Reports are delivered in comma-separated-value (CSV) format. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  "accountIdList": [
    0
  ],
  "omsId": 0,
  "beginTime": "0001-01-01T05:00:00Z",
  "intervalDuration": 0,
  "frequency": {
     "Options": [
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annual"
    ]
  },
}
```

Where:

String	Value
AccountIdList	**integer array.** Comma-separated integers; each element is an account ID whose transaction activity will be reported on. All accounts must be from the same OMS.
OMSId	**integer.** The Order Management System on which the accounts named in the list reside.
beginTime	**string.** The time from which the transaction activities will be reported, in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
intervalDuration	**integer.** The length of time prior to the run time that the report covers, in Microsoft ticks format. For example, 90 days. Whenever the report runs, it looks back 90 days.
frequency	**string.** How often the report will run. One of: 0 OnDemand 1 Hourly 2 Daily 3 Weekly 4 Monthly 5 Annually

Response

Similar objects are returned for **Generate~Report** and **Schedule~Report** calls.

```
{
  "RequestingUser": 0,
  "OMSId": 0,
  "reportFlavor": {
     "Options": [
      "TradeActivity",
      "Transaction",
      "Treasury"
    ]
  },
  "createTime": "0001-01-01T05:00:00Z",
  "initialRunTime": "0001-01-01T05:00:00Z",
  "intervalStartTime": "0001-01-01T05:00:00Z",
  "intervalEndTime": "0001-01-01T05:00:00Z",
  "RequestStatus": {
     "Options": [
      "Submitted",
      "Validating",
      "Scheduled",
      "InProgress",
      "Completed",
      "Aborting",
      "Aborted",
      "UserCancelled",
      "SysRetired",
      "UserCancelledPending"
    ]
  },
  "ReportFrequency": {
     "Options": [
      "onDemand",
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annually"
    ]
  },
  "intervalDuration": 0,
  "RequestId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "lastInstanceId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "accountIds": [
    0
  ],
}
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the transaction activity report. This confirms the ID of the authenticated user who made the request by returning it as part of the response.
OMSId	**integer.** The ID of the Order Management System on which the transaction activity report will be run.
reportFlavor	**string.** The type of report to be generated. One of: TradeActivity Transaction Treasury The *reportFlavor* string confirms the nature of the call.

Table continued on page 151

Table continued from page 150

createTime	**string.** The time and date on which the request for the transaction activity report was made, in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
initialRunTime	**string.** The time and date at which the transaction activity report was first run, in ISO 8601 format and UTC time zone. See ""Time– and Date-Stamp Formats" on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
requestStatus	**string.** The status of the request for the transaction activity report. See "Request Status" on page 10. Each request returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPending
ReportFrequency	**string.** When the report runs. OnDemand Hourly Daily Weekly Monthly Annually
intervalDuration	**long integer.** The period that the report covers relative to the run date. The call specifies a start time and an *intervalDuration* in the form of Microsoft ticks. (See "Time– and Date-Stamp Formats" on page 8.) For example, say that you schedule a weekly report with a 90-day *intervalDuration* value. *intervalDuration* represents the backward-looking period of the report. When the report runs again in a week's time, it again looks back 90 days — but now those 90 days are offset by a week from the first report.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management System.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report.
accountIds	**integer a**rray. A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTradeActivityReport, **GenerateTransactionActivityReport**, **GenerateTreasuryActivityReport**, **GetUserReportTickets**, **GetUserReportWriterResultRecords, ScheduleTradeActivityReport**, **ScheduleTreasuryActivityReport**.

ScheduleTreasuryActivityReport

Schedules a series of treasury activity reports for a list of accounts on a single Order Management System, starting at a specific date/time, and covering a specific time interval. The reports will run periodically until canceled.

The Treasury Activity Report itself is delivered as a comma-separated-value (CSV) file. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  "accountIdList": [
    0
  ],
  "omsId": 0,
  "beginTime": "0001-01-01T05:00:00Z",
  "intervalDuration": 0,
  "frequency": {
     "Options": [
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annual"
    ]
  },
}
```

Where:

String	Value
AccountIdList	**integer array.** Comma-separated integers; each element is an account ID whose treasury activity will be reported on. All accounts must be from the same OMS.
OMSId	**integer.** The Order Management System on which the accounts named in the list reside.
beginTime	**string.** The time from which the treasury activities will be reported, in ISO 8601 format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
intervalDuration	**integer.** The length of time prior to the run time that the report covers, in Microsoft ticks format. For example, 90 days. Whenever the report runs, it looks back 90 days.
frequency	**string.** How often the report will run. One of: 0 OnDemand 1 Hourly 2 Daily 3 Weekly 4 Monthly 5 Annually

Response

Similar objects are returned for **Generate~Report** and **Schedule~Report** calls.

```
{
  "RequestingUser": 0,
  "OMSId": 0,
  "reportFlavor": {
     "Options": [
      "TradeActivity",
      "Transaction",
      "Treasury"
    ]
  },
  "createTime": "0001-01-01T05:00:00Z",
  "initialRunTime": "0001-01-01T05:00:00Z",
  "intervalStartTime": "0001-01-01T05:00:00Z",
  "intervalEndTime": "0001-01-01T05:00:00Z",
  "RequestStatus": {
     "Options": [
      "Submitted",
      "Validating",
      "Scheduled",
      "InProgress",
      "Completed",
      "Aborting",
      "Aborted",
      "UserCancelled",
      "SysRetired",
      "UserCancelledPending"
    ]
  },
  "ReportFrequency": {
     "Options": [
      "onDemand",
      "Hourly",
      "Daily",
      "Weekly",
      "Monthly",
      "Annually"
    ]
  },
  "intervalDuration": 0,
  "RequestId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "lastInstanceId": "AAAAAAAAAAAAAAAAAAAAAA==",
  "accountIds": [
    0
  ],
}
```

Where:

String	Value
RequestingUser	**integer.** The User ID of the person requesting the treasury activity report. This confirms the ID of the authenticated user who made the request by returning it as part of the response.
OMSId	**integer.** The ID of the Order Management System on which the treasury activity report will be run.
reportFlavor	**string.** The type of report to be generated. One of: TradeActivity Transaction Treasury The *reportFlavor* string confirms the nature of the call.

Table continued on page 155

Table continued from page 154

createTime	**string.** The time and date on which the request for the treasury activity report was made, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
initialRunTime	**string.** The time and date at which the treasury activity report was first run, in ISO 8601 format and UTC time zone. See ““Time– and Date-Stamp Formats” on page 8.
intervalStartTime	**string.** The start of the period that the report will cover, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
intervalEndTime	**string.** The end of the period that the report will cover, in ISO 8601 format and UTC time zone. See “Time– and Date-Stamp Formats” on page 8.
requestStatus	**string.** The status of the request for the treasury activity report. See “Request Status” on page 10. Each request returns one of: Submitted Validating Scheduled InProgress Completed Aborting Aborted UserCancelled SysRetired UserCancelledPending
ReportFrequency	**string.** When the report runs. OnDemand Hourly Daily Weekly Monthly Annually
intervalDuration	**long integer.** The period that the report covers relative to the run date. The call specifies a start time and an *intervalDuration* in the form of Microsoft ticks. (See “Time– and Date-Stamp Formats” on page 8.) For example, say that you schedule a weekly report with a 90-day *intervalDuration* value. *intervalDuration* represents the backward-looking period of the report. When the report runs again in a week’s time, it again looks back 90 days — but now those 90 days are offset by a week from the first report.
RequestId	**string.** The ID of the original request. Request IDs are long strings unique within the Order Management System.
lastInstanceId	**string.** For scheduled reports, the report ID of the most recent previously run report.
accountIds	**integer a**rray. A comma-delimited array of account IDs whose trades are reported in the trade activity report.

See Also

GenerateTradeActivityReport, **GenerateTransactionActivityReport**, **GenerateTreasuryActivityReport**, **GetUserReportTickets**, **GetUserReportWriterResultRecords,** **ScheduleTradeActivityReport**, **ScheduleTransactionActivityReport**.

Tickers and Feeds

GetL2Snapshot

No authentication required — trading venue operator may control access

Provides a current Level 2 snapshot of a specific instrument trading on an Order Management System to a user-determined market depth. For more information on Level 1 and Level 2 information, see "Level 1 and Level 2 Market Information" on page 3.

The Level 2 snapshot allows the user to specify the level of market depth information on either side of the bid and ask.

Note: *Depth* in this call is "depth of market," the number of buyers and sellers at greater or lesser prices in the order book for the instrument.

Request

```
{
  "OMSId": 1,
  "InstrumentId": 1,
  "Depth": 100
 }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the instrument is traded.
InstrumentId	**integer.** The ID of the instrument that is the subject of the snapshot.
Depth	**integer.** Depth of the market — the number of buyers and sellers at greater or lesser prices in the order book for the instrument. Defaults to 100.

Response

The response is a single object for one specific instrument. The *Level2UpdateEvent* contains the same data, but is sent by the OMS when trades occur. A user must subscribe to *Level2UpdateEvents*.

```
{
  "MDUpdateID": 0,
  "Accounts": 0,
  "ActionDateTime": 635872032000000000,
  "ActionType": {
    "Options": [
      "New",
      "Update",
      "Delete"
    ]
  "LastTradePrice": 0,
  "Orders": 0,
  "Price": 0,
  "ProductPairCode": 0,
  "Quantity": 0,
  "Side": 0,
}
```

Where:

String	Value
MDUpdateID	**integer.** Market Data Update ID. This sequential ID identifies the order in which the update was created.
Accounts	**integer.** The number of accounts that have orders at this price level.
ActionDateTime	**string.** *ActionDateTime* identifies the time and date that the snapshot was taken or the event occurred, in POSIX format X 1000 (milliseconds since 1 January 1970). See “Time– and Date-Stamp Formats” on page 8.
ActionType	**string.** L2 information provides price data. This value shows whether this data is *new*, an *update*, or a *deletion*. One of: New Update Delete
LastTradePrice	**real.** The price at which the instrument was last traded.
Orders	**integer.** The number of orders at this price point. Whether it is a Buy or Sell order is shown by *Side*, below.
Price	**real.** Bid or Ask price for the *Quantity* (see *Quantity* below).
ProductPairCode	**integer.** *ProductPairCode* is the same number and used for the same purpose as *InstrumentID*. The two are completely equivalent in value. *InstrumentId* 47 = *ProductPairCode* 47.
Quantity	**real.** Quantity available at a given Bid or Ask price (see *Price* above).
Side	**integer.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)

See Also

SubscribeLevel1, SubscribeLevel2, UnsubscribeLevel1, UnsubscribeLevel2

GetTickerHistory

No authentication required

Requests a ticker history (high, low, open, close, volume, bid, ask, ID) of a specific instrument from a given date forward to the present. You will need to format the returned data per your requirements.

Request

```
{
 "InstrumentId": 1,
 "FromDate": // POSIX-format date and time
}
```

Where:

String	Value
InstrumentId	**long integer.** The ID of a specific instrument. The Order Management System and the default Account ID of the logged-in user are assumed.
FromDate	**long integer.** Oldest date from which the ticker history will start, in POSIX format and UTC time zone. The report moves toward the present from this point. See ""Time– and Date-Stamp Formats" on page 8.

Response

The response returns an array of arrays dating from the *FromDate* value of the request. The data are returned oldest-date first. The data returned in the arrays are not labeled. For example, a single returned array element might look like this:

```
[
  1501604532000,
  2792.73,
  2667.95,
  2687.01,
  2700.81,
  242.61340767,
  0,
  2871,
  0
]
```

…and with comments applied to identify the data being returned (comments are not part of the response):

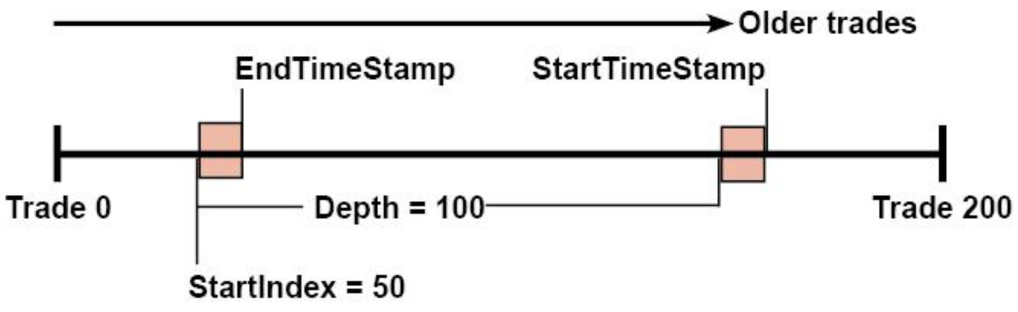
```
[
  1501604532000, // UTC Date/Time in milliseconds since 1/1/1970
  2792.73,      // High
  2667.95,      // Low
  2687.01,      // Open
  2700.81,      // Close
  242.61340767,// Volume
  0,       // Inside bid price
  2871,    // Inside ask price
  0      // Instrument ID
]
```

See Also

SubscribeTicker, UnsubscribeTicker

GetTradesHistory

Retrieves a list of trades for the specified account, order ID, user, instrument, or starting and ending time stamp. The returned list begins at start index *i*, where *i* is an integer identifying a specific trade in reverse order; that is, the most recent trade has an index of 0. "Depth" is the count of trades to report backwards from *StartIndex*.



Caution: You must coordinate *StartIndex, Depth, StartTimeStamp,* and *EndTimeStamp* to retrieve the historical information you need. As the diagram shows, it is possible to specify values (for example, *EndTimeStamp* and *Depth*) that can exclude information you may want (the red areas).

The owner of the trading venue determines how long to retain order history before archiving.

Note: In this call, "Depth" refers not to the depth of the order book, but to the count of trades to report.

Request

All values in the call other than OMSId are optional.

```
{
  "OMSId": 0,
  "AccountId": 0,
  "InstrumentId": 0,
  "TradeId": 0,
  "OrderId": 0,
  "UserId": 0,
  "StartTimestamp": 0,
  "EndTimestamp": 0,
  "Depth": 0,
  "StartIndex": 0,
  "ExecutionId": 0,
}
```

Where:

String	Value
OMSId	**Integer.** The ID of the Order Management System on which the trades took place. If no other values are specified, returns the trades associated with the default account for the logged-in user on this Order Management System.

Table continued on page 164

Table continued from 163

AccountId	**Integer.** The account ID that made the trades. The logged-in user must be associated with this account, although other users also can be associated with the account. If no account ID is supplied, the system assumes the default account for the logged-in user.
InstrumentId	**long integer.** The ID of the instrument whose history is reported. If no instrument ID is included, the system returns trades for all instruments associated with the account and OMS.
TradeId	**integer.** The ID of a specific trade. If specified, the call can return multiple states for a single trade.
OrderId	**integer.** The ID of the order resulting in the trade. If specified, the call returns all trades associated with the order.
UserId	**integer.** The ID of the logged-in user. If not specified, the call returns trades associated with the users belonging to the default account for the logged-in user of this OMS.
StartTimeStamp	**long integer.** The historical date and time at which to begin the trade report, in POSIX format and UTC time zone. If not specified, reverts to the start date of this account on the trading venue. See "Time– and Date-Stamp Formats" on page 8.
EndTimeStamp	**long integer.** Date at which to end the trade report, in POSIX format and UTC time zone. If not specified, uses the current time. See ""Time– and Date-Stamp Formats" on page 8.
Depth	**integer.** In this case, the count of trades to return, counting from the *StartIndex*. If not specified, returns all trades between *BeginTimeStamp* and *EndTimeStamp,* beginning at *StartIndex.*
StartIndex	**integer.** The starting index into the history of trades, from 0 (the most recent trade) and moving backwards in time. If not specified, defaults to 0.
ExecutionId	**integer.** The ID of the individual buy or sell execution. If not specified, returns all.

Response

The response returns an array, one element for each trade.

```
[
  {
    {
    "TradeTimeMS": 0,
    "Fee": 0,
    "FeeProductId": 0,
    "OrderOriginator": 0,
    "OMSId": 0,
    "ExecutionId": 0,
    "TradeId": 0,
    "OrderId": 0,
    "AccountId": 0,
    "SubAccountId": 0,
    "ClientOrderId": 0,
    "InstrumentId": 0,
    "Side": {
       "Options": [
        "Buy",
        "Sell",
        "Short",
        "Unknown"
       ]
```

Code continued on page 165

Code continued from 164

```
              },
              "Quantity": 0,
              "RemainingQuantity": 0,
              "Price": 0,
              "Value": 0,
              "TradeTime": 0,
              "CounterParty": "",
              "OrderTradeRevision": 0,
              "Direction": {
                 "Options": [
                  "NoChange",
                  "UpTick",
                  "DownTick"
                ]
              },
              "IsBlockTrade": false,
            },
          ]
```

Where:

String	Value
TradeTimeMS	**long integer.** The time at which the trade took place, reported in Microsoft ticks format and UTC time zone. See "Time– and Date-Stamp Formats" on page 8.
Fee	**real.** The fee that applied to this trade, if any.
FeeProductId	**integer.** The ID of the product in which the fee is denominated.
OrderOriginator	**integer.** The ID of the user who entered the order on your side of the trade.
OMSId	**integer.** The ID of the Order Management System on which the trade took place.
ExecutionId	**integer.** The ID of your sell or buy side portion of the execution, individually.
TradeId	**integer.** The ID of the overall trade.
OrderId	**integer.** The ID of the order that resulted in the trade.
AccountId	**integer.** The ID of the account under which the trade was executed.
SubAccountId	**integer.** Not currently used.
ClientOrderId	**long integer.** A user-assigned ID for the order (like a purchase-order number assigned by a company). *ClientOrderId* defaults to 0.
InstrumentId	**long integer.** The ID of the instrument being traded.
Side	**string.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
Quantity	**real.** The quantity of the instrument being traded.
RemainingQuantity	**real.** Any quantity remaining in the order after this trade.

Table continued on page 166

Table continued from 165

Price	**real.** The unit price of the order.
Value	**real.** The overall value of the trade — price X quantity.
TradeTime	**long integer.** Time at which the trade took place, in POSIX format and UTC time zone.
CounterParty	**long integer.** Shows 0.
OrderTradeRevision	**integer.** The ID of any trade revision that took place for the trade.
Direction	**string.** Effect of this trade on the market. One of: Nochange UpTick DownTick
IsBlockTrade	**Boolean.** Returns *true* if the trade was a reported trade; *false* otherwise.

See Also

GenerateTradeActivityReport, GetAccountTrades, ScheduleTradeActivityReport, SubscribeTrades, UnsubscribeTrades

SubscribeAccountEvents

Subscribes the user to notifications about the status of account-level events: orders, trades, position updates, deposits, and withdrawals for a specific account on the Order Management System (OMS). The subscription reports all events associated with a given account; there is no filter at the call level to subscribe to some events and not others.

Account event information is supplied in comma-separated-value (CSV) format. For specific CSV formatting information, see the APEX Extract CSV Data Dictionary, available from Coinext.

Request

```
{
  "AccountId":  1,
  "OMSId":  1
}
```

Where:

String	Value
AccountId	integer. The ID of the account for the logged-in user.
OMSId	integer. The ID of the Order Management System to which the account belongs.

Response

```
{
"Subscribe":  true
}
```

Where:

String	Value
Subscribe	**Boolean.** A successful subscription returns true; otherwise, false.

See The Events on page 168

The Events

When you call **SubscribeAccountEvents**, you subscribe to the following list of events. The Order Management System may supply them at irregular intervals; software must listen for these events. The system sends each of these events in a message frame. See ""Message Frame" on page 1.

AccountPositionEvent

Trigger: The balance in your account changes.

```
{
  "OMSId":4, //The OMSId. [Integer]
  "AccountId":4, // account id number. [Integer]
  "ProductSymbol":"BTC",
    //The Product Symbol for this balance message.
  [String] "ProductId":1,
    //The Product Id for this balance message. [Integer]
  "Amount":10499.1,
    //The total balance in the account for the specified product.
  [Dec] "Hold": 2.1,
    //The total amount of the balance that is on hold. Your available
    //balance for trading and withdraw is (Amount - Hold). [Decimal]
  "PendingDeposits":0,
    //Total Deposits Pending for the specified product.
  [Decimal] "PendingWithdraws":0,
    //Total Withdrawals Pending for the specified product.
  [Decimal] "TotalDayDeposits":0,
    //The total 24-hour deposits for the specified product. UTC.
  [Dec] "TotalDayWithdraws":0
    //The total 24-hour withdraws for the specified product. UTC [Dec]
}
```

CancelAllOrdersRejectEvent

Trigger: All orders for your account are rejected.

```
{
  "OMSId": 1, // OMS ID [Integer]
  "AccountId": 4, // ID of the account being tracked [Integer]
  "InstrumentId": 0,
    // ID of the instrument in the order [Long Integer]
  "Status": "Rejected", // Accepted/Rejected [String]
  "RejectReason": "Instrument not found."
    // Reason for rejection [String]
}
```

CancelOrderRejectEvent

Trigger: Your order is canceled.

```
{
  "OMSId": 1, //OMS Id [Integer] Always 1
  "AccountId": 4, //Your Account ID. [Integer]
  "OrderId": 1,
    //The Order ID from your Cancel request. [64 Bit Integer]
  "OrderRevision": 0,
    //The Revision of the Order, if any was found. [64 Bit
  Integer] "OrderType": "Unknown", // See "Order Types" on page 7
  "InstrumentId": 1,
    // The InstrumentId from your Cancel request. [Integer]
  "Status": "Rejected", //Always "Rejected" [String]
  "RejectReason": "Order Not Found"
    //A message describing the reason for the rejection. [String]
}
```

CancelReplaceOrderRejectEvent

Trigger: Your order is rejected even if a cancel-replace order was placed.

```
{
  "OMSId": 1, // ID of the OMS [integer]
  "AccountId": 4, // ID of the account [integer]
  "OrderId": 9342, // The ID of the rejected order [integer]
  "ClientOrderId": 1234, // The client-supplied order ID [long integer]
  "LimitPrice": 99.1, // The limit price of the order.
  "OrderIdOCO": 0,
    // The ID of the other ordre to cancel if this is executed.
  "OrderType": "Limit", // See "Order Types" on page 7.
  "PegPriceType": "Bid", // Where to peg the stop/trailing
  order. "OrderIdToReplace": 9333,
    // The ID of the order being cancelled and replaced.
  "InstrumentId": 1, // ID of the instrument traded in the
  order. "ReferencePrice": 99.1, // used internally.
  "Quantity": 1.0, // Quantity of the replacement order
  "Side": "Buy", // Side of the order: Buy, Sell, Short (future)
  "StopPrice":0, // The price at which to execute the new order.
  "TimeInForce":"GTC", // Period when new order can be executed.
  "Status":"Rejected", // Status of the order – always "rejected"
  "RejectReason":"Order Not Found" // Reason the order was rejected.
}
```

MarketStateUpdate

Trigger: The market state is altered administratively.

```
{
  "ExchangeId":1,      // Exchange Id [Integer]
  "VenueAdapterId":1,  // Internal [Integer]
  "VenueInstrumentId":1, // Instrument Id on a specific venue
  [Integer] "Action":"ReOpen",
    // Market State Action [String] Values are
    // "Pause", "Resume", "Halt",
  "ReOpen" "PreviousStatus":"Stopped",
    // Previous Market Status for Instrument [String] Values are
    // "Running", "Paused", "Stopped",
  "Starting" "NewStatus":"Running",
    // Market Status for Instrument [String] Values are
    // "Running", "Paused", "Stopped", "Starting"
  "ExchangeDateTime":"2016-04-21T21:48:22Z"
    // ISO 8601 format UTC time zone
}
```

NewOrderRejectEvent

Trigger: An order associated with your account is rejected.

```
{
  "OMSId": 1, //OMS Id [Integer] Always 1
  "AccountId": 4, //Your Account Id [Integer]
  "ClientOrderId": 1234, //Your Client Order Id [64 Bit Integer]
  "Status": "Rejected", //Always "Rejected"
  "RejectReason": "No More Market"
    //A message describing the reason for the reject.
}
```

OrderStateEvent

Trigger: The status changes for an order associated with your account.

```
{
  "Side":"Sell",
    // The side of your order. [String] Values are "Sell",
    // "Buy", "Short"
  "OrderId": 9849, //The Server-Assigned Order Id. [64-bit Integer]
  "Price": 97, //The Price of your order. [Decimal]
```

Code continued on page 170

Code continued from page 169

```
“Quantity”:1,
  // The Quantity (Remaining if partially or fully executed) of
  // your order. [Decimal]
“Instrument”:1, // The InstrumentId your order is for.
[Integer] “Account”:4, // Your AccountId [Integer]
“OrderType”:”Limit”,
  // The type of order. [String] Values are “Market”, “Limit”,
  // “StopMarket”, “StopLimit”, “TrailingStopMarket”, and
  // “TrailingStopLimit”
“ClientOrderId”:0, // Your client order id. [64-bit Integer]
“OrderState”:”Working”, // The current state of the order. [String]
  // Values are “Working”, “Rejected”, “FullyExecuted”, “Canceled”,
  // “Expired”
“ReceiveTime”:0,     // Timestamp in POSIX format
“OrigQuantity”:1,    // The original quantity of your order. [Decimal]
“QuantityExecuted”:0, // The total executed quantity. [Decimal]
“AvgPrice”:0,        // Avergage executed price. [Decimal]
“ChangeReason”:”NewInputAccepted”
  // The reason for the order state change. [String] Values are
  // “NewInputAccepted”, “NewInputRejected”, “OtherRejected”,
  // “Expired”, “Trade”, SystemCanceled BelowMinimum”,
  // “SystemCanceled NoMoreMarket”, “UserModified”
```

OrderTradeEvent

Trigger: An order associated with your account results in a trade.

```
{
  “OMSId”:1,           //OMS Id [Integer]
  “TradeId”:213,       /Trade Id [64-bit Integer]
  “OrderId”:9848,      //Order Id [64-bit Integer]
  “AccountId”:4,       //Your Account Id [Integer]
  “ClientOrderId”:0,   //Your client order id. [64-bit Integer]
  “InstrumentId”:1,    //Instrument Id [Integer]
  “Side”:”Buy”,
    //[String] Values are “Buy”, “Sell”, “Short” (future)
  “Quantity”:0.01,     //Quantity [Decimal]
  “Price”:95,          //Price [Decimal]
  “Value”:0.95,        //Value [Decimal]
  “TradeTime”:635978008210426109,
    // TimeStamp in Microsoft ticks
  format “ContraAcctId”:3,
    // The Counterparty of the trade. The counterparty is always
    // the clearing account. [Integer]
  “OrderTradeRevision”:1, //Usually 1
  “Direction”:”NoChange” //”Uptick”, “Downtick”, “NoChange”
}
```

PendingDepositUpdate

Trigger: Deposit pending on your account.

```
{
  “AccountId”: 4,  // Your account id number. [Integer]
  “AssetId”: 1,    // The ProductId of the pending deposit. [Integer]
  “TotalPendingDepositValue”: 0.01
    //The value of the pending deposit. [Decimal]
}
  “Requires2FA”: false,
  “TwoFAType”: “”,
  “TwoFAToken”: “”,
}
```

See Also

SubscribeLevel1, SubscribeLevel2, SubscribeTicker, SubscribeTrades, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTicker, UnsubscribeTrades

SubscribeLevel1

No authentication required — trading venue operator may control access

Retrieves the latest Level 1 Ticker information and then subscribes the user to ongoing Level 1 market data event updates for one specific instrument. For more information about Level 1 and Level 2, see “Level 1 and Level 2 Market Information” on page 3. The **SubscribeLevel1** call responds with the Level 1 response shown below. The OMS then periodically sends *Leve1UpdateEvent* information when best bid/best offer issues in the same format as this response, until you send the **UnsubscribeLevel1** call.

Request

You can identify the instrument with its ID or with its market symbol (string).

```
{
  “OMSId”:  1,
  “InstrumentId”: 0
}
```

Or

```
{
  “OMSId”:  1,
  “Symbol”: “BTCUSD”}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instrument trades.
InstrumentId	**integer.** The ID of the instrument you’re tracking. *Conditionally optional.*
Symbol	**string.** The symbol of the instrument you’re tracking. *Conditionally optional.*

Response

The **SubscribeLevel1** response and *Level1UpdateEvent* both provide the same information.

```
{
  “OMSId”: 1,
  “InstrumentId”: 1,
  “BestBid”: 0.00,
  “BestOffer”: 0.00,
  “LastTradedPx”: 0.00,
  “LastTradedQty”: 0.00,
  “LastTradeTime”: 635872032000000000,
  “SessionOpen”: 0.00,
  “SessionHigh”: 0.00,
  “SessionLow”: 0.00,
  “SessionClose”: 0.00,
  “Volume”: 0.00,
  “CurrentDayVolume”: 0.00,
  “CurrentDayNumTrades”: 0,
  “CurrentDayPxChange”: 0.0,
  “Rolling24HrVolume”: 0.0,
  “Rolling24NumTrades”: 0.0,
  “Rolling24HrPxChange”: 0.0,
```

Code continued on page 172

Code continued from page 171

```
                    “TimeStamp”: 635872032000000000,
                  }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instrument trades.
InstrumentId	**integer.** The ID of the instrument being tracked.
BestBid	**real.** The current best bid for the instrument.
BestOffer	**real.** The current best offer for the instrument.
LastTradedPx	**real.** The last-traded price for the instrument.
LastTradedQty	**real.** The last-traded quantity for the instrument.
LastTradeTime	**long integer.** The time of the last trade in POSIX format X 1000 (milliseconds since 1 January 1970). See “Time– and Date-Stamp Formats” on page 8.
SessionOpen	**real.** Opening price. In markets with openings and closings, this is the opening price for the current session; in 24-hour markets, it is the price as of UTC Midnight.
SessionHigh	**real.** Highest price during the trading day, either during a true session (with opening and closing prices) or UTC midnight to UTC midnight.
SessionLow	**real.** Lowest price during the trading day, either during a true session (with opening and closing prices) or UTC midnight to UTC midnight.
SessionClose	**real.** The closing price. In markets with openings and closings, this is the closing price for the current session; in 24-hour markets, it is the price as of UTC Midnight.
Volume	**real.** The unit volume of the instrument traded, either during a true session (with openings and closings) or in 24-hour markets, the period from UTC Midnight to UTC Midnight.
CurrentDayVolume	**real.** The unit volume of the instrument traded either during a true session (with openings and closings) or in 24-hour markets, the period from UTC Midnight to UTC Midnight.
CurrentDayNumTrades	**integer.** The number of trades during the current day, either during a true session (with openings and closings) or in 24-hour markets, the period from UTC Midnight to UTC Midnight.
CurrentDayPxChange	**real.** Current day price change, either during a true trading session or UTC Midnight to UTC midnight.
Rolling24HrVolume	**real.** Unit volume of the instrument during the past 24 hours, regardless of time zone. Recalculates continuously.
Rolling24HrNumTrades	**integer.** Number of trades during the past 24 hours, regardless of time zone. Recalculates continuously.
Rolling24HrPxChange	**real.** Price change during the past 24 hours, regardless of time zone. Recalculates continuously.

Table continued on page 173

Table continued from page 172

TimeStamp	**long integer.** The time this information was provided, in POSIX format X 1000 (milliseconds since 1 January 1970). See “Time– and Date-Stamp Formats” on page 8.

See Also

SubscribeAccountEvents, SubscribeLevel2, SubscribeTicker, SubscribeTrades, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTicker, UnsubscribeTrades

SubscribeLevel2

No authentication required — trading venue operator may control access

Retrieves the latest Level 2 Ticker information and then subscribes the user to Level 2 market data event updates for one specific instrument. Level 2 allows the user to specify the level of market depth information on either side of the bid and ask. For more information about Level 1 and Level 2, see "Level 1 and Level 2 Market Information" on page 3. The **SubscribeLevel2** call responds with the Level 2 response shown below. The OMS then periodically sends *Level2UpdateEvent* information in the same format as this response until you send the **UnsubscribeLevel2** call.

Note: *Depth* in this call is "depth of market," the number of buyers and sellers at greater or lesser prices in the order book for the instrument.

Request

You can identify the instrument either by ID or by market symbol.

```
{
  "OMSId":  1,
  "InstrumentId": 0
  "Depth":  10
}
```

or

```
{
  "OMSId":  1,
  "Symbol": "BTCUSD"
  "Depth":  10
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instrument trades.
InstrumentId	**integer.** The ID of the instrument you're tracking. *Conditionally optional.*
Symbol	**string.** The symbol of the instrument you're tracking. *Conditionally optional.*
Depth	**integer.** The depth of the order book. The example request returns 10 price levels on each side of the market.

Response

The response is a single object (for one specific instrument). The *Level2UpdateEvent* contains the same data, but is sent by the OMS when trades occur.

```
{
  "MDUpdateID": 0,
  "Accounts": 0,
  "ActionDateTime": 635872032000000000,
```

Code continued on page 176

Code continued from page 175

```
    "ActionType": {
      "Options": [
        "New",
        "Update",
        "Delete"
      ]
    "LastTradePrice": 0,
    "Orders": 0,
    "Price": 0,
    "ProductPairCode": 0,
    "Quantity": 0,
    "Side": 0,
  }
```

Where:

String	Value
MDUpdateID	**integer.** Market Data Update ID. This sequential ID identifies the order in which the update was created.
Accounts	**integer.** The number of accounts that have orders at this price level.
ActionDateTime	**string.** *ActionDateTime* identifies the time and date that the snapshot was taken or the event occurred, in POSIX format X 1000 (milliseconds since 1 January 1970). See "Time– and Date-Stamp Formats" on page 8.
ActionType	**string.** L2 information provides price data. This value shows whether this data is *new*, an *update*, or a *deletion*. One of: New Update Delete
LastTradePrice	**real.** The price at which the instrument was last traded.
Orders	**integer.** The number of orders at this price point. Whether it is a Buy or Sell order is shown by *Side*, below.
Price	**real.** Bid or Ask price for the *Quantity* (see *Quantity* below).
ProductPairCode	**integer.** *ProductPairCode* is the same number and used for the same purpose as *InstrumentID*. The two are completely equivalent in value. *InstrumentId* 47 = *ProductPairCode* 47.
Quantity	**real.** Quantity available at a given Bid or Ask price (see *Price* above).
Side	**integer.** One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeTicker, SubscribeTrades, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTicker, UnsubscribeTrades

SubscribeTicker

No authentication required — trading venue operator may control access

Subscribes a user to a Ticker Market Data Feed for a specific instrument and interval. **SubscribeTicker** sends a response object as described below, and then periodically returns a *TickerDataUpdateEvent* that matches the content of the response object.

Request

```
{
  "OMSId": 1,
  "InstrumentId": 1,
  "Interval": 60,
  "IncludeLastCount": 100
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System
InstrumentId	**long integer.** The ID of the instrument whose information you want to track.
Interval	**integer.** Specifies in seconds how frequently to obtain ticker updates. Default is 60 — one minute.
IncludeLastCount	**integer.** The limit of records returned in the ticker history. The default is 100.

Response

The response returns an array of objects, each object an unlabeled, comma-delimited set of numbers. The *Open* price and *Close* price are those at the beginning of the tick — the Interval time subscribed to in the request.

A typical response might look like this:

```
[[1510719222970.21,6943.51,6890.27,6898.41,6891.16,0,6890.98,6891.98,1,
1510718681956.34]],
```

Here are the values in order with an explanation:

```
[
  {
    "EndDateTime": 0, // POSIX format
    "HighPX": 0,
    "LowPX": 0,
    "OpenPX": 0,
    "ClosePX": 0,
    "Volume": 0,
    "Bid": 0,
    "Ask": 0,
    "InstrumentId": 1,
    "BeginDateTime": 0 // POSIX format
  }
]
```

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeLevel2, SubscribeTrades, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTicker, UnsubscribeTrades

SubscribeTrades

Subscribes an authenticated user to the Trades Market Data Feed for a specific instrument. Each trade has two sides: *Buy* and *Sell*.

SubscribeTrades returns the response documented here for your immediate information, then periodically sends the *OrderTradeEvent* documented in **SubscribeAccountEvents**.

Request

```
{
  "OMSId": 1,
  "InstrumentId": 1,
  "IncludeLastCount": 100
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the instrument is traded.
InstrumentId	**long integer.** The ID of the instrument whose trades will be reported.
IncludeLastCount	**integer.** Specifies the number of previous trades to retrieve in the immediate snapshot. Default is 100.

Response

The response returns an array of trades. Both sides of each trade are described.

```
[
  {
    {
      "OMSId": 0,
      "TradeID": 0,
      "ProductPairCode": 0,
      "Quantity": 0,
      "Price": 0,
      "Order1": 0,
      "Order2": 0,
      "TradeTime": "0001-01-01T05:00:00Z",
      "Direction": {
          "Options":   [
            "NoChange",
            "UpTick",
            "DownTick"
          ]
        },
      "TakerSide": 0,
      "Side1AccountId": 0,
      "Side2AccountId": 0,
      "Order1Side": {
          "Options":   [
            "Buy",
            "Sell",
```

Code continued on page 180

Code continued from page 179

```
                              "Short",
                              "Unknown"
                            ]
                          },
                        "Order2Side": {
                            "Options":    [
                              Buy",
                              "Sell",
                              "Short",
                              "Unknown"
                            ]
                          },
                        "BlockTrade": false,
                        "Order1ClientId": 0,
                        "Order2ClientId": 0,
                      },
                    }
                  ]
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System where the instrument to be tracked is traded.
TradeID	**integer.** The ID of this trade.
ProductPairCode	**integer.** *ProductPairCode* is the same number and used for the same purpose as *InstrumentID*. The two are completely equivalent in value. *InstrumentId* 47 = *ProductPairCode* 47.
Quantity	**real.** The quantity of the instrument traded.
Price	**real.** The price at which the instrument traded.
Order1	**integer.** The ID of one of the orders that resulted in the trade.
Order2	**integer.** The ID of the other order that resulted in the trade.
TradeTime	**long integer.** The time at which the trade took place. UTC time. See "Time– and Date-Stamp Formats" on page 8.
Direction	**string.** Effect of the trade on the instrument's market price. One of: 0 NoChange 1 UpTick 2 DownTick
TakerSide	**integer.** Which side of the trade took liquidity? One of: 0 Buy 1 Sell The maker side of the trade provides liquidity by placing the order on the book (this can be a buy or a sell order). The other side takes the liquidity. It, too, can be buy-side or sell-side.
Side1AccountId	**integer.** The account ID of the 1-side of the trade.
Side2AccountId	**integer.** The account ID of the 2-side of the trade.

Table continued on page 181

Table continued from page 180

Order1Side	**string.** The side taken by order 1 of the trade. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
Order2Side	**string.** The side taken by order 2 of the trade. One of: 0 Buy 1 Sell 2 Short (reserved for future use) 3 Unknown (error condition)
BlockTrade	**Boolean.** Was this a privately negotiated trade that was reported to the OMS? A private trade returns *true*; otherwise *false*. Default is *false*.
Order1ClientId	**long integer.** The client-supplied order ID of the 1-side client.
Order2ClientId	**long integer.** The client-supplied order ID of the 2-side client.

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeLevel2, SubscribeTicker, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTicker, UnsubscribeTrades

UnsubscribeLevel1

No authentication required

Unsubscribes the user from a Level 1 Market Data Feed subscription.

Request

```
{
 "OMSId": 1,
 "InstrumentId": 1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the user has subscribed to a Level 1 market data feed.
InstrumentId	**long integer.** The ID of the instrument being tracked by the Level 1 market data feed.

Response

```
{
  "result": true,
  "errormsg": null,
  "errorcode":0,
  "detail": null
}
```

Where:

String	Value
result	**Boolean.** A successful receipt of the unsubscribe request returns *true*; and unsuccessful receipt (an error condition) returns *false*.
errormsg	**string.** A successful receipt of the unsubscribe request returns null; the *errormsg* parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeLevel2, SubscribeTicker, SubscribeTrades, UnsubscribeLevel2, UnsubscribeTicker, UnsubscribeTrades

UnsubscribeLevel2

No authentication required

Unsubscribes the user from a Level 2 Market Data Feed subscription..

Request

```
  "OMSId": 1,
  "InstrumentId": 1
 }
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the user has subscribed to a Level 2 market data feed.
InstrumentId	**long integer.** The ID of the instrument being tracked by the Level 2 market data feed.

Response

```
{
  "result": true,
  "errormsg": null,
  "errorcode":0,
  "detail": null
}
```

Where:

String	Value
result	**Boolean.** A successful receipt of the unsubscribe request returns *true*; and unsuccessful receipt (an error condition) returns *false*.
errormsg	**string.** A successful receipt of the unsubscribe request returns null; the *errormsg* parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeLevel2, SubscribeTicker, SubscribeTrades, UnsubscribeLevel1, UnsubscribeTicker, UnsubscribeTrades

UnsubscribeTicker

No authentication required

Unsubscribes a user from a Ticker Market Data Feed

Request

```
[
 “OMSId”: 1,
 “InstrumentId”: 1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the user has subscribed to a ticker market data feed.
InstrumentId	**long integer.** The ID of the instrument being tracked by the ticker market data feed.

Response

```
{
  “result”: true,
  “errormsg”: null,
  “errorcode”:0,
  “detail”: null
}
```

Where:

String	Value
result	**Boolean.** A successful receipt of the unsubscribe request returns *true*; and unsuccessful receipt (an error condition) returns *false*.
errormsg	**string.** A successful receipt of the unsubscribe request returns null; the *errormsg* parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeLevel2, SubscribeTicker, SubscribeTrades, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTrades

UnsubscribeTrades

No authentication required

Unsubscribes a user from the Trades Market Data Feed.

Request

```
[
 "OMSId": 1,
 "InstrumentId": 1
}
```

Where:

String	Value
OMSId	**integer.** The ID of the Order Management System on which the user has subscribed to a trades market data feed.
InstrumentId	**long integer.** The ID of the instrument being tracked by the trades market data feed.

Response

```
{
  "result": true,
  "errormsg": null,
  "errorcode":0,
  "detail": null
}
```

Where:

String	Value
result	**Boolean.** A successful receipt of the unsubscribe request returns *true*; and unsuccessful receipt (an error condition) returns *false*.
errormsg	**string.** A successful receipt of the unsubscribe request returns null; the *errormsg* parameter for an unsuccessful request returns one of the following messages: Not Authorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
errorcode	**integer.** A successful receipt of the unsubscribe request returns 0. An unsuccessful receipt returns one of the *errorcodes* shown in the *errormsg* list.
detail	**string.** Message text that the system may send. Usually *null*.

See Also

SubscribeAccountEvents, SubscribeLevel1, SubscribeLevel2, SubscribeTicker, SubscribeTrades, UnsubscribeLevel1, UnsubscribeLevel2, UnsubscribeTicker