

Table of Contents

Background Information	1
Contents Common to Many API Calls	7

Registration and authentication

RegisterNewUser	13
WebAuthenticateUser	15
Authenticate2FA.....	17
LogOut	19
ResetPassword	21

User Information Calls

GetAvailablePermissionList.....	25
GetUserConfig.....	27
GetUserInfo.....	29
GetUserPermissions.....	31
RemoveUserConfig	33
SetUserConfig	35
SetUserInfo	37

Order-handling calls

CancelAllOrders	41
CancelOrder.....	43
CancelQuote.....	45
CancelReplaceOrder	47
CreateQuote	51
GetAccountInfo.....	53
GetAccountPositions	57
GetAccountTrades	59
GetAccountTransactions.....	63
GetInstrument.....	67
GetInstruments.....	71
GetOpenOrders	75
GetOpenQuotes	79
GetOrderFee.....	85
GetOrderHistory	87
GetOrderHistoryByOrderId	91
GetOrdersHistory	95

GetOrderStatus.....	101
GetProduct.....	105
GetProducts.....	107
GetUserAccounts.....	109
GetUserAccountInfos	111
ModifyOrder	115
SendOrder	117
UpdateQuote	121

Reports

CancelUserReport	125
GenerateTradeActivityReport	127
GenerateTransactionActivityReport.....	131
GenerateTreasuryActivityReport	135
GetUserReportTickets	139
GetUserReportWriterResultRecords	143
ScheduleTradeActivityReport	145
ScheduleTransactionActivityReport.....	149
ScheduleTreasuryActivityReport	153

Tickers and Feeds

GetL2Snapshot	159
GetTickerHistory	161
GetTradesHistory.....	163
SubscribeAccountEvents.....	167
SubscribeLevel1	171
SubscribeLevel2	175
SubscribeTicker	177
SubscribeTrades.....	179
UnsubscribeLevel1	183
UnsubscribeLevel2	185
UnsubscribeTicker.....	187
UnsubscribeTrades	189

Deposits and Withdrawals

CreateDepositTicket.....	193
CreateWithdrawTicket.....	197

GetAllDepositRequestInfoTemplates	199
GetAllDepositTickets	203
GetAllWithdrawTickets	207
GetDepositInfo.....	213
GetDepositRequestInfoTemplate	215
GetDepositTicket.....	217
GetWithdrawTemplate.....	221
GetWithdrawTemplateTypes.....	223
GetWithdrawTicket.....	225
GetWithdrawTickets.....	229
UpdateDepositTicket	235
UpdateWithdrawTicket	239

Index

Index	245
-------------	-----

Background Information

This section provides important information about the exchange software..

Message Frame

Wrap all calls to the software in a JSON-formatted frame object. Responses from the software are similarly wrapped.

```
{
  "m": 0,
  "i": 0,
  "n": "function_name",
  "o": "payload"
}
```

Where:

String	Value
<i>m</i> message type	integer. The type of the message: 0 request 1 reply 2 subscribe to event 3 event 4 unsubscribe from event 5 error
<i>i</i> sequence number	long integer. The sequence number identifies an individual request, or request-and-response pair, to your application. A non-zero sequence number is required, but the numbering scheme you use is up to you. No arbitrary sequence numbering scheme is enforced. Best practices: A client-generated API call (of message types 0, 2, and 4) should: Carry an even sequence number. Begin at the start of each user session. Be unique within each user session. Begin with 2 (2, 4, 6, 8). Message types 1 (reply), 3 (event), and 5 (error) are generated by the server. These messages echo the sequence number of the message to which they respond. See Example, below..
<i>n</i> function name	string. The function name is the name of the function that you are calling or that the server responds to. The server echoes your call. See Example, below.
<i>o</i> payload	<i>Payload</i> is a JSON-formatted string containing the data being sent with the message. Payload may consist of request parameters (string-value pairs) or response parameters.

Note: You can send the string-value pairs inside the payload in any order; the server controls the order of the response.

Example

When sending a request in the frame to the software using JavaScript, a call looks like:

```
var frame =
{
  "m":0,
  "i":0,
  "n":"function name",
  "o":""
};

var requestPayload =
{
  "Parameter1":"Value",
  "Parameter2":0
};
frame.o = json.Stringify(requestPayload);
//Stringify escapes the payload's quotation marks automatically.
WS.Send(json.Stringify(frame)); //WS.Send escapes the frame.
```

When receiving a frame from the software, use the frame to determine the context, and then unwrap the content:

```
var frame = json.Parse(wsMessage);

if (frame.m == 1) //message of type reply
{
  //This is a Reply
  if (frame.n == "WebAuthenticateUser")
  {
    var LoginReply = json.Parse(frame.o);
    if (LoginReply.Authenticated)
    {
      var user = LoginReply.User;
    }
  }
}
```

Note: If not using JSON Stringify, escape quotation marks.

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

Where:

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

Table continued from page 2

errormsg	string. A successful receipt of the call returns null; the <i>errormsg</i> 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 <i>errorcodes</i> shown in the <i>errormsg</i> list.
detail	string. Message text that the system may send. The content of this parameter is usually <i>null</i> .

Authenticated and Public Calls

All API calls require that the user be logged-in and authenticated — with the following exceptions that can be called without prior authentication:

Authenticate2FA	GetTickerHistory
WebAuthenticateUser	SubscribeLevel1
LogOut	SubscribeLevel2
GetInstrument	SubscribeTicker
GetInstruments	UnsubscribeLevel1
GetProduct	UnsubscribeLevel2
GetProducts	UnsubscribeTicker

However, the individual trading venue operator can control access to Ticker History and the Subscribe calls. Calls that require no authentication or that are venue-controlled are indicated underneath each call name.

Level 1 and Level 2 Market Information

The software allows users to subscribe to Level 1 and Level 2 market information.

- ↔ **Level 1 information** consists of real-time bid and ask quotes for a specific instrument ID or instrument symbol. Level 1 provides last trade information; session open, high, low, and close (if the venue operates in sessions; 24-hour information if not); current, and rolling information.
- ↔ **Level 2 information** also consists of real-time bid and ask quotes for a specific instrument ID or instrument symbol as does Level 1, but allows the user to specify the level of market depth information on either side of the bid and ask, and to see quantities.

Access to Level 1 and Level 2 market information subscriptions is controlled by the trading venue operator. API calls that involve Level 1 or Level 2 information include: **GetL2Snapshot**, **SubscribeLevel1**, **UnsubscribeLevel1**, **SubscribeLevel2**, and **UnsubscribeLevel2**.

Modules

The software consists of several modules that include the Order Management System, the matching engine, and the Asset Manager. During installation, each is assigned an ID.

Background Information

The *Order Management System* — or OMS — is the mechanism that manages access to the trading venue. The OMS controls permissions, accounts, and users. The OMS ID must be specified in most calls.

The order book resides on the *matching engine*. A trading venue is a combination of OMS and matching engine that creates a place to access the market and trade. A venue maintains its order book and matching engine, and may access several Order Management Systems.

The *Asset Manager* controls the deposit and withdrawal of the funds belonging to an account. These funds can be denominated in any product that the trading venue allows.

Permissions

An individual user may have multiple accounts. Similarly, multiple users may have trading privileges through a single account. There may be users who have trading access to one set of accounts that overlap (but do not fully duplicate) another user's set of accounts. There may be a many-to-many relationship where two or more users have access to a set of accounts.

The use case for this kind of "joint tenancy" is an active trading desk where a specific individual may not always be present. If User A will not be present, User B can monitor and trade on the market. User A may wish to cancel his pending trades for a specific account or instrument, but not those of his trading partner under the same account or for the same instrument.

Permissions handle the rules that determine what a user can access and do with orders, cancellations, and other tasks in a trading venue. Most permissions encompass tasks such as trading, depositing, or making withdrawals; but permission can be set for each API call for each individual in the venue. This controls what a user can do.

The system administrator sets up permissions on the OMS when the user joins the trading venue, and only the system administrator can change them. A full discussion of permissions is not part of this document.

Products and Instruments

In the 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). The software thinks of that exchange as explicit.

Quotes and Orders

The 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 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 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	

Background Information

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**, **GetOrderHistoryById**, **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

The 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). The 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 the software, 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](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 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**.

Registration and authentication

RegisterNewUser

Creates and registers a new user account on an Order Management System. A single account can have many users; a single user can be associated with many accounts. See “Permissions” on page 4.

Request

```
{
  "userInfo": {
    "UserName": "testusername",
    "Email": "abc@ap.com",
    "PasswordHash": "pword",
  },
  "OperatorId": 1,
  "UserConfig": []
}
```

Where:

String	Value
userInfo	User Info object. See <i>userInfo</i> object, below.
OperatorId	long integer . The unique ID number of the operator of the OMS.
UserConfig	Array of string-value pairs as required by the venue operator. See <i>UserConfig</i> array, below.

userInfo object:

String	Value
UserName	string . Readable name of the user. Example: JSmith.
Email	string . Email address of the user being registered.
PasswordHash	string . The password assigned to the new user, in clear.
AffiliatedId	integer . The ID of an affiliated organization, if the new user comes from an affiliated link. Set to 0 if no affiliated organization. This is an optional field.

UserConfig array:

UserConfig holds an arbitrary array of string/value pairs named *Name* and *Value* (in other calls referred to as *Key* and *Value*) as required by the venue operator. When creating a new user, the *UserConfig* array may be empty. There are separate **set-** and **get-** **userconfig** calls.

```
[
  {
    "Name": "billingZip",
```

Code continued on page 14

RegisterNewUser

Code continued from page 13

```
        "Value": "19312",
        "Name": "billingCity",
        "Value": "Berwyn",
        "Name": "billingState",
        "Value": "Pennsylvania"
    }
]
or
[
    {
        "Name": "2FAType",
        "Value": "Google"
    }
]
```

Note: In **RegisterNewUser**, the configuration strings are called "Name." In **SetUserConfig** and **GetUserConfig** (and other places where configuration strings are used), the strings are called "Key." Value is called "Value" in all cases.

Response

```
{
  "UserID": 1
}
```

Where:

String	Value
UserID	integer. The Order Management System echoes the User ID assigned by the venue operator..

See Also

WebAuthenticateUser

WebAuthenticateUser

No authentication required

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

Request

```
{
  "UserName": "UserName",
  "Password": "Password"
}
```

Where:

String	Value
UserName	string. The name of the user, for example, jsmith.
Password	string. The user password. The user logs into a specific Order Management System via Secure Socket Layer (SSL and HTTPS).

Response

Unsuccessful response:

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

Where:

String	Value
Authenticated	Boolean. The default response is <i>false</i> for an unsuccessful authentication.

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

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

Where:

User Object:

String	Value
Authenticated	Boolean. The response is <i>true</i> for a successful authentication.

WebAuthenticateUser

Table continued from page 15

SessionToken	string. <i>SessionToken</i> 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](#), [LogOut](#)

Authenticate2FA

No authentication required

Completes the second part of a two-factor authentication by sending the authentication token from the non-software 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 **AuthenticateUser**. 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 <i>true</i> . Unsuccessful returns <i>false</i> .
SessionToken	string. The <i>SessionToken</i> 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 <i>SessionToken</i> instead of repeating the full two-factor authentication process. A session lasts one hour after last-detected activity or until logout.

Authenticate2FA

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

```
{  
  "SessionToken": "YourSessionToken"  
}
```

See Also

WebAuthenticateUser, LogOut

Logout ends the current websocket session.

Request

There is no payload for a *Logout* request.

```
{ }
```

Response

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

Where:

String	Value
result	Boolean. A successful logout returns true; and unsuccessful logout (an error condition) returns false.
errmsg	string. A successful logout returns null; the errmsg 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 errmsg list.
detail	string. Message text that the system may send. Usually null.

See Also:

[AuthenticateUser](#), [Authenticate2FA](#)

ResetPassword

ResetPassword is a two-step process. The first step calls **ResetPassword** with the user's username. The Order Management System then sends an email to the user's registered email address. The email contains a reset link. Clicking the link sends the user to a web page where he can enter a new password.

Note: Passwords must be longer than eight characters, with at least one case change (upper-to-lower/ lower-to-upper) and one numeral. Venue operators may impose additional rules.

Request

```
{
  "UserName": "UserName",
}
```

Where:

String	Value
UserName	string. The name of the user, for example, jsmith.

Response

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

Where:

String	Value
result	Boolean. Returns <i>true</i> if the UserName is valid; <i>false</i> if not. See "Standard Response Object and Common Error Codes" on page 2 for an explanation of the other string/value pairs.

See Also

[RegisterNewUser](#)

User Information Calls

GetAvailablePermissionList

Retrieves a comma-separated array of all permissions that can be assigned to a user.

An administrator or superuser can set permissions for each user on an API-call by API-call basis, to allow for highly granular control. Common permission sets include *Trading*, *Deposit*, and *Withdrawal* (which allow trading, deposit of funds, and account withdrawals, respectively); or *AdminUI*, *UserOperator*, and *AccountOperator* (which allow control of the Order Management System, set of users, or an account). See “Permissions” on page 4 for more information, but a complete discussion of permissions and their scope is beyond this API guide.

Request

The request for **GetAvailablePermissionList** does not require a `UserId`. It returns a list of all permissions available.

```
{ }
```

Response

A successful response returns an alphabetically sorted list of all permissions that can be assigned by an administrator or superuser.

```
[  
  "AccountReadOnly" ,  
  "AddAccount" ,  
  "AddBalanceReconciliationInfo" ,  
  "AddDepositTicketAttachment" ,  
  "AddEditAccountBadge" ,  
  "AddEditExchange" ,  
  "AddEditExchangeInstrument" ,  
  "AddEditOMS" ,  
  "AddEditOperatorEmail" ,  
  "AddInstrument" ,  
  "AddOperator" ,  
  "AddOperatorOms" ,  
  ...  
]
```

See Also

GetUserPermissions

GetUserConfig

GetUserConfig returns the list of key/value pairs set by the **SetUserConfig** call and associated with a user record. A trading venue can use *Config* strings to store custom information or compliance information with a user record.

Note: In **RegisterNewUser** (and only in **RegisterNewUser**), the *key* identifier of the *config* string is called *name*.

Request

```
{
  "UserId": 1,
  "UserName": "jsmith",
}
```

Where:

String	Value
UserId	integer. The ID of the user whose permission information will be returned. A user can only retrieve his own permissions; an administrator can retrieve information about the permissions of others.
UserName	string. The login name of the user; "jsmith."

Response

A successful call to **GetUserConfig** returns an array of the current list of *Config* key/value pairs. Because they are set during the **RegisterNewUser** and **SetUserConfig** calls, *Config* key/value pairs are unique to their trading venues.

An unsuccessful call to **GetUserConfig** returns an error code. See "Standard Response Object and Common Error Codes" on page 2.

```
[
  "Street": "Hillside Road",
  "Office Number": 158,
  "Mobile Phone": "1-702-555-1212",
  "City": "Las Vegas",
]
```

See Also

RegisterNewUser, **SetUserConfig**

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? <i>True</i> if yes; <i>false</i> if no. Defaults to <i>false</i> .
AccountId	integer. The ID of the default account with which the user is associated.

Table continued on page 30

GetUserInfo

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 is not associated with an affiliated organization.
ReferrerId	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. <i>True</i> if the user must use two-factor authentication; <i>false</i> if the user does not need to use two-factor authentication. Defaults to <i>false</i> .
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](#)

GetUserPermissions

Retrieves an array of permissions for the logged-in user. Permissions can be set only by an administrator or superuser.

An administrator or superuser can set permissions for each user on an API-call by API-call basis, to allow for highly granular control. Common permission sets include *Trading*, *Deposit*, and *Withdrawal* (which allow trading, deposit of funds, and account withdrawals, respectively); or *AdminUI*, *UserOperator*, and *AccountOperator* (which allow control of the Order Management System, set of users, or an account). See “Permissions” on page 4 for more information, but a complete discussion of permissions and their scope is beyond this API guide.

Request

```
{
  "UserId": 1,
}
```

Where:

String	Value
UserId	integer. The ID of the user whose permission information will be returned. A user can only retrieve his own permissions; an administrator can retrieve information about the permissions of others.

Response

A successful response returns an array of permission strings. An unsuccessful response returns an error code. See “Standard Response Object and Common Error Codes” on page 2 for more information about error codes.

```
[
  "Withdraw",
  "Deposit",
  "Trading"
]
```

See Also

[GetAvailablePermissionList](#)

RemoveUserConfig

RemoveUserConfig deletes a single key/value *Config* pair from a user record. A trading venue uses *Config* strings to store custom information or compliance information with a user's record.

Note: In **RegisterNewUser** (and only in **RegisterNewUser**), the *key* identifier of the config strings is called *name*.

Request

A *Key* is always a string. The call deletes a single *Key* and its paired *Value*, even though the value is not specified in the request.

```
{
  "UserId": 1,
  "UserName": "jsmith",
  "Key": "Street Name",
}
```

Where:

String	Value
UserId	integer. The ID of the user from whose record you're deleting the custom key/value pair.
UserName	string. The name of the user from whose record you're deleting the custom key/value pair.
Key	string. The name of the key/value pair to delete..

Response

A response from the server (even a successful one) indicates only that the Order Management System has received the request; not that the key/value pairs have been removed. The **GetUserConfig** call returns the current list of *Config* key/value pairs; use it to verify that the correct pairs have been successfully deleted.

A successful response from the OMS returns a *true* result, null *errmsg*, and a 0 *errorcode*; an unsuccessful response returns a *false* result, a string *errmsg*, and a numeric *errorcode*. For more information see "Message Frame" on page 1 and "Standard Response Object and Common Error Codes" on page 2.

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

RemoveUserConfig

Where:

String	Value
result	Boolean. If the call has been successfully received by the Order Management System, result is <i>true</i> ; otherwise, it is <i>false</i> .
errmsg	string. A successful receipt of the call returns <i>null</i> ; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. The content of this parameter is usually null.

See Also

[GetUserConfig](#), [RegisterNewUser](#), [SetUserConfig](#)

SetUserConfig

SetUserConfig adds an array of one or more arbitrary key/value pairs to a user record. A trading venue can use *Config* strings to store custom information or compliance information with a user's record.

Note: In **RegisterNewUser** (and only in **RegisterNewUser**), the *key* identifier of the config strings is called *name*.

Request

A *Key* is always a string, but the associated *Value* of the *Key* can be of any data type.

```
{
  "UserId": 1,
  "UserName": "jsmith",
  "Config": [
    "Key": "Street Name",
    "Value": "Hillside Road",
    "Key": "Suite Number",
    "Value": 158,
    ...
  ]
}
```

Where:

String	Value
UserId	integer. The ID of the user to whose record you're adding the custom key/value pairs.
UserName	string. The name of the user to whose record you're adding the custom key/value pairs.
Config	array of key/value pairs. "Key" is always a string; but the associated <i>Value</i> of <i>Key</i> can be of any data type.

Response

A response from the server (even a successful one) indicates only that the Order Management System has received the request; not that the key/value pairs have been stored. The **GetUserConfig** call returns the current list of *Config* key/value pairs; use it to verify that your new pairs have been successfully stored.

A successful response from the OMS returns a *true* result, null *errmsg*, and a 0 *errorcode*; an unsuccessful response returns a *false* result, a string *errmsg*, and a numeric *errorcode*. For more information see "Message Frame" on page 1 and "Standard Response Object and Common Error Codes" on page 2.

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

SetUserConfig

Where:

String	Value
result	Boolean. If the call has been successfully received by the Order Management System, result is <i>true</i> ; otherwise, it is <i>false</i> .
errmsg	string. A successful receipt of the call returns <i>null</i> ; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. The content of this parameter is usually null.

See Also

[GetUserConfig](#), [RegisterNewUser](#)

SetUserInfo

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

Request

```
{
  "UserId": 1,
  "UserName": "John Smith",
  "Password": "password",
  "Email": "email@company.com",
  "EmailVerified": true,
  "AccountId": 1,
  "Use2FA": false,
}
```

Where:

String	Value
UserId	integer. The ID of the user; set by the system when the user registers.
UserName	string. User's name; "John Smith."
Password	string. User's password.
Email	string. User's email address.
EmailVerified	Boolean. Send <i>true</i> if you have verified the user's email; send <i>false</i> if you have not verified the email address. Default is <i>false</i> .
AccountId	integer. The ID of the default account with which the user is associated. A user may be associated with more than one account, and more than one user may be associated with a single account. An admin or superuser can specify additional accounts.
Use2FA	Boolean. Set to <i>true</i> if this user must use two-factor authentication; set to <i>false</i> if this user does not need to use two-factor authentication. Default is <i>false</i> .

Response

A successful response echoes the settings in the request, and provides additional information about the user information from the database. 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,
```

Code continued on page 38

SetUserInfo

Code continued from page 37

```
"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. Contains a GUID — a globally unique ID string — during the time that a new user has been sent a registration email and before the user clicks the confirmation link.
EmailVerified	Boolean. Has your organization verified this email as correct and operational? <i>True</i> if yes; <i>false</i> if no. Defaults to <i>false</i> .
AccountId	integer. The ID of the default account with which the user is associated.
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 is 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. <i>True</i> if the user must use two-factor authentication; <i>false</i> if the user does not need to use two-factor authentication. Defaults to <i>false</i> .
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](#), [GetUserInfo](#),
[GetUserConfig](#), [SetUserConfig](#),

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	Acc't	Instr	
37	14	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.

CancelAllOrders

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 <i>true</i> ; otherwise, it is <i>false</i> .
errormsg	string. A successful receipt of the call returns null; the <i>errormsg</i> 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 <i>errorcodes</i> shown in the <i>errormsg</i> list.
detail	string. Message text that the system may send. The content of this parameter is usually <i>null</i> .

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

CancelOrder

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 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.

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

Where:

String	Value
OMSIId	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,
    \"errmsg\": \"\",
    \"errorcode\": 0,
    \"detail\": \"\",
  }",
  "askresult": "{
    \"result\": true,
```

Code continued on page 46

CancelQuote

Code continued from page 45

```
        "errmsg": "",  
        "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.
errmsg	string. A successful receipt of the cancellation returns null; the errmsg 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 errmsg 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"
    ]
  }
}
```

CancelReplaceOrder

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. <i>ClientOrderId</i> 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 <i>PegPriceType</i> 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 <i>OrderIdOCO</i> 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 <i>ClientOrderId</i> value from the request.
OrigOrderId	integer. Echoes <i>OrderIdToReplace</i> , 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

CancelReplaceOrder

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 the software. Only registered market participants or market makers may quote.

Request

```
{
  "OMSIId": 0,
  "AccountId": 0,
  "InstrumentId": 0,
  "Bid": 0,
  "BidQty": 0,
  "Ask": 0,
  "AskQty": 0,
}
```

Where:

String	Value
OMSIId	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 <i>AccountId</i> , 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,
    "errmsg": "",
    "errorcode": 0,
    "detail": "",
  }",
  "AskQuoteId": 0,
  "AskResult": "{
    "result": true,
```

Code continued on page 52

CreateQuote

Code continued from page 51

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

Where:

String	Value
BidQuoteld	integer. The ID of the bid quote returned by the Order Management System.
BidResult	string. Returns a standard response object for Bid.
AskQuoteld	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.
errmsg	string. A successful receipt of the request returns null; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

See Also

[CancelQuote](#), [GetOpenQuotes](#), [UpdateQuote](#)

GetAccountInfo

Returns detailed information about one specific account belonging to the authenticated user and existing on a specific Order Management System.

Request

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

Where:

String	Value
OMSId	integer. The ID of the Order Management System on which the account exists.
AccountId	integer. The ID of the account on the Order Management System for which information will be returned.
AccountHandle	string. <i>AccountHandle</i> is a unique user-assigned name that is checked at create time by the Order Management System. Alternate to Account ID.

Response

```
{
  "OMSID": 0,
  "AccountId": 0,
  "AccountName": "",
  "AccountHandle": "",
  "FirmId": "",
  "FirmName": "",
  "AccountType": {
    "Options": [
      "Asset",
      "Liability",
      "ProfitLoss"
    ]
  },
  "FeeGroupID": 0,
  "ParentID": 0,
  "RiskType": {
    "Options": [
      "Unknown",
      "Normal",
      "NoRiskCheck",
      "NoTrading"
    ]
  },
  "VerificationLevel": 0,
  "FeeProductType": {
    "Options": [
      "BaseProduct",
      "SingleProduct"
    ]
  }
}
```

GetAccountInfo

Code continued from page 53

```

    ]
  },
  "FeeProduct": 0,
  "RefererId": 0,
  "SupportedVenueIds": [
    0
  ],
}

```

Where:

String	Value
OMSId	integer. The ID of the Order Management System on which the account resides.
AccountId	integer. The ID of the account for which information was requested.
AccountName	string. A non-unique name for the account assigned by the user.
AccountHandle	string. <i>AccountHandle</i> is a unique user-assigned name that is checked at create time by the Order Management System to assure its uniqueness.
FirmId	string. An arbitrary identifier assigned by a trading venue operator to a trading firm as part of the initial company, user, and account set up process. For example, Smith Financial Partners might have the ID SMFP.
FirmName	string. A longer, non-unique version of the trading firm's name; for example, Smith Financial Partners.
AccountType	string. The type of the account for which information is being returned. One of: Asset Liability ProfitLoss Responses for this string/value pair for Market Participants are almost exclusively Asset.
FeeGroupID	integer. Defines account attributes relating to how fees are calculated and assessed. Set by trading venue operator.
ParentID	integer. Reserved for future development.
RiskType	string. One of: Unkown (an error condition) Normal NoRiskCheck NoTrading Returns Normal for virtually all market participants. Other types indicate account configurations assignable by the trading venue operator.
VerificationLevel	integer. Verification level ID (how much verification does this account require) defined by and set by the trading venue operator for this account.
FeeProductType	string. One of: BaseProduct SingleProduct Trading fees may be charged by a trading venue operator. This value shows whether fees for this account's trades are charged in the product being traded (<i>BaseProduct</i> , for example BitCoin) or whether the account has a preferred fee-paying product (<i>SingleProduct</i> , for example USD) to use in all cases and regardless of product being traded.

Table continued on page 55

Table continued from page 54

FeeProduct	integer . The ID of the preferred fee product, if any. Defaults to 0.
RefererId	integer . Captures the ID of the person who referred this account to the trading venue, usually for marketing purposes.
SupportedVenuels	integer array . Comma-separated array. Reserved for future expansion.

See Also

[GetUserAccounts](#), [GetUserAccountInfos](#), [SubscribeAccountEvents](#)

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
  }
]
```

GetAccountPositions

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 <i>PendingWithdraws</i> 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

GetAccountTrades

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

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

GetAccountTransactions

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

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

GetInstrument

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 <i>standard</i> , 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 <i>VenueId</i> .
VenueId	integer. The ID of the trading venue on which the instrument trades, if not this venue. See <i>VenueInstrumentId</i> .
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, <i>SortIndex</i> 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 <i>true</i> ; otherwise defaults to <i>false</i> .
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

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"
        ]
      }
    }
  ]
}
```

GetInstruments

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 <i>VenueId</i> .
VenueId	integer. The ID of the trading venue on which the instrument trades, if not this venue. See <i>VenueInstrumentId</i> .
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 <i>true</i> ; otherwise defaults to <i>false</i> .
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,
  }
]
```

GetOpenOrders

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 <i>OrderId</i> is unique in each Order Management System.
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 <i>DisplayQuantity</i> 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). <i>ClientOrderId</i> 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 <i>OrigQuantity</i> 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 the original order. This number is also appended to CancelReplaceOrder . See CancelReplaceOrder .
EnteredBy	integer. User ID of the person who entered the order.
IsQuote	Boolean. <i>True</i> if the open order is a quote; <i>false</i> 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, <i>RejectReason</i> holds the reason for the rejection.
IsLockedIn	Boolean. <i>True</i> if both parties to a block trade agree that one party will report the trade for both. Otherwise <i>false</i> .
OMSId	integer. ID of the Order Management System on which the order was placed.

GetOpenOrders

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 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,
    "OMSIId": 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

GetOpenQuotes

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 <i>DisplayQuantity</i> 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). <i>ClientOrderId</i> defaults to 0.
OrderState	string. One of: Unknown Working Rejected Canceled Expired FullyExecuted An open quote will probably have an <i>OrderState</i> 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 <i>true</i> , otherwise <i>false</i> . Default is <i>false</i> .
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. <i>True</i> if both parties to a block trade agree that one party will report the trade for both. Otherwise <i>false</i> .
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.

GetOrderFee

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

GetOrderHistory

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). <i>ClientOrderId</i> 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.
OrigClientId	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

GetOrderHistory

Table continued from page 89

IsQuote	Boolean. If this order is a quote (rather than an order), returns <i>true</i> , otherwise <i>false</i> . Default is <i>false</i> .
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. <i>True</i> if both parties to a block trade agree that one party will report the trade for both. Otherwise <i>false</i> .
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",

```

GetOrderHistoryByOrderId

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 <i>DisplayQuantity</i> 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). <i>ClientOrderId</i> 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.
OrigClientId	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

GetOrderHistoryByOrderId

Table continued from page 93

IsQuote	Boolean. If this order is a quote (rather than an order), returns <i>true</i> , otherwise <i>false</i> . Default is <i>false</i> .
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). <i>ClientOrderId</i> 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.

GetOrdersHistory

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 <i>StartIndex</i> . If not specified, returns all orders between <i>BeginTimeStamp</i> and <i>EndTimeStamp</i> , beginning at <i>StartIndex</i> 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 <i>DisplayQuantity</i> 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

GetOrdersHistory

Table continued from page 97

ClientOrderId	long integer. A user-assigned ID for the order (like a purchase-order number assigned by a company). <i>ClientOrderId</i> 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 <i>Quantity</i> or <i>OrigQuantity</i>).
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.
OrigClientId	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 <i>true</i> , otherwise <i>false</i> . Default is <i>false</i> .
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. <i>True</i> if both parties to a block trade agree that one party will report the trade for both. Otherwise <i>false</i> .
OMSId	integer. The ID of the Order Management System on which the order was created.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOpenQuotes, GetOrderHistory, GetOrderHistoryById, 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",
```

GetOrderStatus

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 <i>DisplayQuantity</i> 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). <i>ClientOrderId</i> 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 (<i>Quantity</i>).
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.
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). The default value is 0.
EnteredBy	integer. The ID of the user who originally entered the order.

Table continued on page 104

GetOrderStatus

Table continued from page 103

IsQuote	Boolean. Returns <i>true</i> if the order is a quote, else returns <i>false</i> . Default is <i>false</i> . 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. <i>True</i> if both parties to a block trade agree that one party will report the trade for both. Otherwise <i>false</i> .
OMSId	integer. ID of the Order Management System on which the trades being reported on occurred.

See Also

CancelAllOrders, CancelOrder, CancelReplaceOrder, GetOpenOrders, GetOpenQuotes, GetOrderHistory, GetOrderHistoryById, 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

```
{
  "OMSIId": 1,
  "ProductId": 1
}
```

Where:

String	Value
OMSIId	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:

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

Where:

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

GetProduct

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 <i>QuantityIncrement</i> . 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 <i>false</i> ; that is, if <i>NoFees</i> is <i>false</i> , fees will be incurred. If <i>NoFees</i> is <i>true</i> , 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

```
{
  "OMSIId": 1,
}
```

Where:

String	Value
OMSIId	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.

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

Where:

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

GetProducts

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 <i>QuantityIncrement</i> . 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 <i>false</i> ; that is, if <i>NoFees</i> is <i>false</i> , fees will be incurred. If <i>NoFees</i> is <i>true</i> , no fees are incurred.

See Also

GetAccountPositions, GetInstrument, GetInstruments, GetProduct

GetUserAccounts

Returns a list of account IDs for a given user. More than one user may be associated with a given account. For more information about accounts and users, see “Permissions” on page 4.

Request

```
{
  "OMSid": 0,
  "UserId": 0,
  "UserName": "",
}
```

Where:

String	Value
OMSid	integer. The Order Management System on which the user has one ore more accounts.
UserId	integer. The ID of the user whose accounts you want to return.
UserName	string. The name of the user.

Response

The response returns list of comma-separated account IDs.

```
{
  0, // a list of account IDs
}
```

See Also

[GetAccountInfo](#), [GetUserAccountInfos](#)

GetUserAccountInfos

Returns a list of account information for all accounts belonging to the specified user.

Request

```
{
  "OMSId": 0,
  "UserId": 0,
  "UserName": "",
}
```

Where:

String	Value
OMSId	integer. The Order Management System on which the user has one ore more accounts.
UserId	integer. The ID of the user whose accounts you want to return.
UserName	string. The name of the user.

Response

Returns a JSON list of account objects one for each account associated with the user.

```
{
  "OMSID": 0,
  "AccountId": 0,
  "AccountName": "",
  "AccountHandle": "",
  "FirmId": "",
  "FirmName": "",
  "AccountType": {
    "Options": [
      "Asset",
      "Liability",
      "ProfitLoss"
    ]
  },
  "FeeGroupID": 0,
  "ParentID": 0,
  "RiskType": {
    "Options": [
      "Unknown",
      "Normal",
      "NoRiskCheck",
      "NoTrading"
    ]
  },
  "VerificationLevel": 0,
  "FeeProductType": {
    "Options": [
      "BaseProduct",
      "SingleProduct"
    ]
  }
}
```

GetUserAccountInfos

Code continued from page 111

```
    },
    "FeeProduct": 0,
    "RefererId": 0,
    "SupportedVenueIds": [
        0
    ],
}
```

Where:

String	Value
OMSID	integer. The ID of the Order Management System on which the account resides.
AccountId	integer. The ID of the account for which information was requested.
AccountName	string. A non-unique name for the account assigned by the user.
AccountHandle	string. <i>AccountHandle</i> is a unique user-assigned name that is checked at create time by the Order Management System.
FirmId	string. An arbitrary identifier assigned by a trading venue operator to a trading firm as part of the initial company, user, and account set up process. For example, Smith Financial Partners might have the ID SMFP.
FirmName	string. A longer, non-unique version of the trading firm's name; for example, Smith Financial Partners.
AccountType	string. The type of the account for which information is being returned. One of: Asset Liability ProfitLoss Responses for this string/value pair for Market Participants are almost exclusively Asset.
FeeGroupID	integer. Defines account attributes relating to how fees are calculated and assessed. Set by trading venue operator.
ParentID	integer. Reserved for future development.
RiskType	string. One of: Unkown (an error condition) Normal NoRiskCheck NoTrading Returns Normal for virtually all market participants. Other types indicate account configurations assignable by the trading venue operator.
VerificationLevel	integer. Verification level ID (how much verification does this account require) defined by and set by the trading venue operator for this account.
FeeProductType	string. One of: BaseProduct SingleProduct Trading fees may be charged by a trading venue operator. This value shows whether fees for this account's trades are charged in the product being traded (<i>BaseProduct</i> , for example BitCoin) or whether the account has a preferred fee-paying product (<i>SingleProduct</i> , for example USD) to use in all cases and regardless of product being traded.

Table continued from page 112

FeeProduct	integer. The ID of the preferred fee product, if <i>SingleProduct</i> is the value of <i>FeeProductType</i> .
RefererId	integer. Captures the ID of the person who referred this account to the trading venue, usually for marketing purposes.
SupportedVenuels	integer array. Comma-separated array. Reserved for future expansion.

See Also

GetAccountInfo, GetUserAccounts,

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

ModifyOrder

Where:

String	Value
result	Boolean. The successful receipt of a modify request returns <i>true</i> ; otherwise, returns <i>false</i> . 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 <i>null</i> ; the <i>errormsg</i> 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 <i>errormsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

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. <i>ClientOrderId</i> 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 <i>DisplayQuantity</i> 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 <i>UseDisplayQuantity</i> to <i>true</i> . See "Display Quantity" on page 8 for more information about how the system uses the <i>DisplayQuantity</i> value.
LimitPrice	real. The price at which to execute the order, if the order is a Limit order.
OrderIdOCO	integer. <i>One Cancels the Other</i> — If this order is order A, <i>OrderIdOCO</i> 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 .

SendOrder

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 <i>PegPriceType</i> 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",  
  "errmsg": "",  
  "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
errmsg	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, GetOrderHistoryById, GetOrdersHistory, GetOrderStatus, ModifyOrder

SendOrder

UpdateQuote

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

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

Request

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

Where:

String	Value
OMSIId	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,
    "errmsg": "",
    "errorCode": 0,
```

UpdateQuote

```
        "detail": "",
    },
    "AskQuoteId": 0,
    "AskResult": "{
        \"result\": true,
        \"errmsg\": \"\",
        \"errorcode\": 0,
        \"detail\": \"\",
    }"
```

Where:

String	Value
BidQuoteId	integer. The ID of the Bid quote being updated.
BidResult	string. Returns a response object for Bid.
AskQuoteId	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 <i>true</i> ; and unsuccessful receipt of the update (an error condition) returns <i>false</i> .
errmsg	string. A successful receipt of the update returns <i>null</i> ; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

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 <i>errormsg</i> list.

Table continued on page 126

CancelUserReport

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.

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. <i>startTime</i> 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. <i>endTime</i> 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"
    ]
  }
}
```

GenerateTradeActivityReport

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 <i>reportFlavor</i> 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 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 <i>Submitted</i> . 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 <i>OnDemand</i> . 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 software calculates the difference between them as <i>intervalDuration</i> . 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 <i>intervalDuration</i> 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 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.
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.

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 or 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. <i>startTime</i> 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. <i>endTime</i> 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

GenerateTransactionActivityReport

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 <i>reportFlavor</i> 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 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 <i>Submitted</i> . 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 <i>OnDemand</i> . 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 software calculates the difference between them as <i>intervalDuration</i> . 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 <i>intervalDuration</i> 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 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, 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.

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 or 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. <i>startTime</i> 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. <i>endTime</i> 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"
    ]
  }
}
```

GenerateTreasuryActivityReport

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 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 <i>OnDemand</i> . 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 software calculates the difference between them as <i>intervalDuration</i> . 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 <i>intervalDuration</i> 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 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, 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"
          ]
        }
      }
    }
  ]
}
```

GetUserReportTickets

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 from page 140

intervalDuration	long integer. The period that the report looks backward relative to the run date. The system calculates <i>intervalDuration</i> between <i>intervalStartTime</i> and <i>intervalEndTime</i> 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 <i>intervalDuration</i> 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 <i>null</i> 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.

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

GetUserReportWriterResultRecords

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.

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 <i>reportFlavor</i> string confirms the nature of the call.

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 <i>intervalDuration</i> 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 <i>intervalDuration</i> value. <i>intervalDuration</i> 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 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.

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 <i>reportFlavor</i> 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 <i>intervalDuration</i> 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 <i>intervalDuration</i> value. <i>intervalDuration</i> 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 array. 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.

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 <i>reportFlavor</i> 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 <i>intervalDuration</i> 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 <i>intervalDuration</i> value. <i>intervalDuration</i> 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 array. 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. <i>ActionDateTime</i> 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 <i>new</i> , an <i>update</i> , or a <i>deletion</i> . 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 <i>Side</i> , below.
Price	real. Bid or Ask price for the <i>Quantity</i> (see <i>Quantity</i> below).
ProductPairCode	integer. <i>ProductPairCode</i> is the same number and used for the same purpose as <i>InstrumentID</i> . The two are completely equivalent in value. <i>InstrumentId</i> 47 = <i>ProductPairCode</i> 47.
Quantity	real. Quantity available at a given Bid or Ask price (see <i>Price</i> 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
]
```

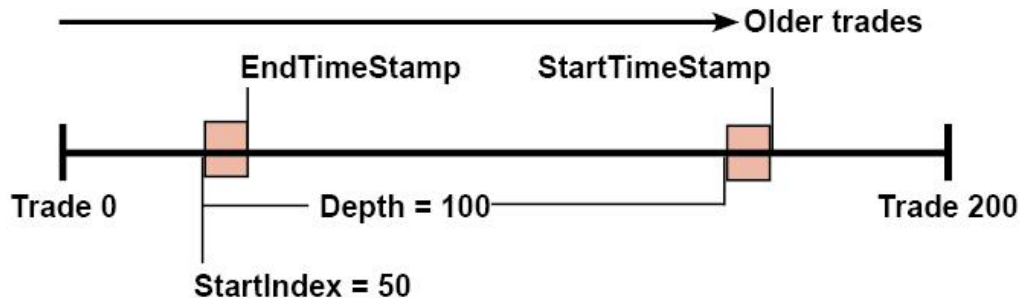
GetTickerHistory

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.

GetTradesHistory

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 <i>StartIndex</i> . If not specified, returns all trades between <i>BeginTimeStamp</i> and <i>EndTimeStamp</i> , beginning at <i>StartIndex</i> .
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 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). <i>ClientOrderId</i> 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

GetTradesHistory

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 <i>true</i> if the trade was a reported trade; <i>false</i> 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.

Request

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

Where:

String	Value
AccountId	integer. The ID of the account for the logged-in user.
OMSIId	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.

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.

```
{
  "OMSIId":4, //The OMSTId. [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.

```
{
  "OMSIId": 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.

```
{
  "OMSIId": 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 order 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]
```

SubscribeAccountEvents

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, // Average 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 *Level1UpdateEvent* 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).

```
{
  "OMSIId": 1,
  "InstrumentId": 0
}

Or

{
  "OMSIId": 1,
  "Symbol": "BTCUSD" }
```

Where:

String	Value
OMSIId	integer. The ID of the Order Management System on which the instrument trades.
InstrumentId	integer. The ID of the instrument you're tracking. <i>Conditionally optional.</i>
Symbol	string. The symbol of the instrument you're tracking. <i>Conditionally optional.</i>

Response

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

```
{
  "OMSIId": 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 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.

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

or

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

Where:

String	Value
OMSIId	integer. The ID of the Order Management System on which the instrument trades.
InstrumentId	integer. The ID of the instrument you're tracking. <i>Conditionally optional.</i>
Symbol	string. The symbol of the instrument you're tracking. <i>Conditionally optional.</i>
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,
```

SubscribeLevel2

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. <i>ActionDateTime</i> 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 <i>new</i> , an <i>update</i> , or a <i>deletion</i> . 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 <i>Side</i> , below.
Price	real. Bid or Ask price for the <i>Quantity</i> (see <i>Quantity</i> below).
ProductPairCode	integer. <i>ProductPairCode</i> is the same number and used for the same purpose as <i>InstrumentID</i> . The two are completely equivalent in value. <i>InstrumentId</i> 47 = <i>ProductPairCode</i> 47.
Quantity	real. Quantity available at a given Bid or Ask price (see <i>Price</i> 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

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

Where:

String	Value
OMSIId	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

SubscribeTrades

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. <i>ProductPairCode</i> is the same number and used for the same purpose as <i>InstrumentId</i> . The two are completely equivalent in value. <i>InstrumentId</i> 47 = <i>ProductPairCode</i> 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 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 <i>true</i> ; otherwise <i>false</i> . Default is <i>false</i> .
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,
  "errmsg": null,
  "errorcode": 0,
  "detail": null
}
```

Where:

String	Value
result	Boolean. A successful receipt of the unsubscribe request returns <i>true</i> ; and unsuccessful receipt (an error condition) returns <i>false</i> .
errmsg	string. A successful receipt of the unsubscribe request returns null; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

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,
  "errmsg": null,
  "errorCode": 0,
  "detail": null
}
```

Where:

String	Value
result	Boolean. A successful receipt of the unsubscribe request returns <i>true</i> ; and unsuccessful receipt (an error condition) returns <i>false</i> .
errmsg	string. A successful receipt of the unsubscribe request returns null; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

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,
  "errmsg": null,
  "errorcode": 0,
  "detail": null
}
```

Where:

String	Value
result	Boolean. A successful receipt of the unsubscribe request returns <i>true</i> ; and unsuccessful receipt (an error condition) returns <i>false</i> .
errmsg	string. A successful receipt of the unsubscribe request returns null; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

See Also

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

UnsubscribeTrades

No authentication required

Unsubscribes a user from the Trades Market Data Feed.

Request

```
[
  "OMSIId": 1,
  "InstrumentId": 1
]
```

Where:

String	Value
OMSIId	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,
  "errmsg": null,
  "errorCode": 0,
  "detail": null
}
```

Where:

String	Value
result	Boolean. A successful receipt of the unsubscribe request returns <i>true</i> ; and unsuccessful receipt (an error condition) returns <i>false</i> .
errmsg	string. A successful receipt of the unsubscribe request returns null; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string. Message text that the system may send. Usually <i>null</i> .

See Also

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

Deposits and Withdrawals

CreateDepositTicket

CreateDepositTicket records a deposit ticket for deposits of fiat money (non-crypto national currencies, for example). Crypto-currencies, such as BitCoin or Monero are handled by a different deposit mechanism described in **GetDepositInfo**.

The ticketing mechanism of the Order Management System tracks deposits and withdrawals, interacting with the Asset Manager. For more information on these modules, see “Modules” on page 3.

Request

Note: The set of *DepositInfo* string/value pairs in the request is not related to information provided by the **GetDepositInfo** call.

```
{
  "AccountId":1,
  "AssetId":2,
  "AssetName": "USD",
  "Amount":60,
  "OMSid":1,
  "RequestUser":5,
  "OperatorId":1,
  "Status":"New",
  "DepositInfo": "{
    \"Full Name\": \"John Smith\",
    \"language\": \"en\",
    \"Bank Name\": \"\",
    \"Comment\": \"initial deposit\"
  }"
```

Where:

String	Value
AccountId	integer. The account into which the deposit is going to be made.
AssetId	integer. The ID of the asset being deposited. Equivalent to product ID. AssetId = ProductId, and uses the same ID numbers.
AssetName	string. The name of the asset being deposited. USD (dollars), BTC (bitcoin), gold, NZD (New Zealand dollars) for example.
Amount	real. The amount of the asset being deposited.
OMSid	integer. The ID of the Order Management System on which the deposit is being made.
RequestUser	integer. The ID of the user making the deposit.
OperatorId	integer. The ID of the trading venue operator.

CreateDepositTicket

Table continued from page 193

Status	<p>string. The current status of the deposit, stated as a string. One of:</p> <ul style="list-style-type: none">0 New1 AdminProcessing2 Accepted3 Rejected4 SystemProcessing5 FullyProcessed6 Failed7 Pending <p>The <i>Status</i> of a newly created deposit ticket should always be <i>New</i>.</p> <hr/> <p>Note: Withdraw tickets have three additional <i>Status</i> values.</p>
DepositInfo	<p>JSON string. See <i>DepositInfo</i> object, below. The <i>DepositInfo</i> JSON string holds specific string/value pairs with information about the deposit.</p>

DepositInfo object:

String	Value
Full Name	string. The name of the user making the deposit.
language	string. A two-letter abbreviation for the language in which the deposit is written — <i>en</i> for English, <i>sp</i> for Spanish, <i>ru</i> for Russian, and so forth. These abbreviations are not enumerated to allow for flexibility.
Bank Name	string. The name of the bank from which the deposit is being made.
Comment	string. A comment by the depositor.

Response

The successful response to **CreateDepositTicket** is a Boolean *true* value and a request code to allow tracking the ticket. To view and confirm ticket contents, use the call **GetDepositTicket**.

```
{
  "success": true,
  "requestcode": "866f21fe-3461-41d1-91aa-5689bc38503f",
}
```

Where:

String	Value
success	Boolean. Returns <i>true</i> if the system has created the deposit ticket; otherwise returns <i>false</i> .
requestcode	string. A globally-unique ID (GUID) that identifies this specific deposit ticket.

An unsuccessful response to **CreateDepositTicket** is a standard response object that includes an error code and error message, as explained in “Standard Response Object and Common Error Codes” on page 2.

See Also

GetAllDepositTickets, GetDepositInfo, GetDepositTicket, UpdateDepositTicket.

CreateWithdrawTicket

Initiates the withdrawal of funds from an account on the trading venue, and transfers them to an external account through a third-party “account provider.”

Note: **CreateWithdrawTicket** requires administrator approval before the withdrawal occurs unless you have configured auto-approval settings.

Request

Part of the process of withdrawing an asset (product) is to specify the withdrawal template that sends the asset to the correct destination, usually an account provider. The content of templates varies from account provider to account provider. Get a list of templates available to you by calling **GetWithdrawTemplateTypes**. For more information on templates, see “Deposit and Withdraw Templates” on page 9. The code below shows a sample template.

```
{
  "OMSId":1,
  "AccountId":4,
  "ProductId":1,
  "Amount":1.0,
  "TemplateType":"ToExternalBitcoinAddress",
  "TemplateForm":
    "{
      \"TemplateType\":\"ToExternalBitcoinAddress\",
      \"Comment\":\"\",
      \"ExternalAddress\":\"54123214\"
    }"
```

Where:

String	Value
OMSId	integer. ID of the Order Management System on which the withdrawal is being made.
AccountId	integer. The ID of the account from which to make the withdrawal.
ProductId	integer. The ID of the product being withdrawn, for example US Dollars. See “Products and Instruments” on page 4 for more information about the nature of a product. When you call CreateDepositTicket to add funds to an account, <i>ProductId</i> is referred to as <i>AssetId</i> . Both <i>ProductId</i> and <i>AssetId</i> refer to the same thing.
Amount	real. The amount of the product being withdrawn from the account.
TemplateType	string. The name of the withdrawal template that controls the destination of the asset being withdrawn. To get a list of withdrawal templates that are available to you, call GetWithdrawTemplateTypes .
TemplateForm	Object. See the <i>TemplateForm</i> object below.

CreateWithdrawTicket

TemplateForm object. The content of the *TemplateForm* object varies from account provider to account provider, depending on the asset being withdrawn and the identity of the account provider. This is an example of one *TemplateForm*.

String	Value
TemplateType	string . The name of the template.
Comment	string . A comment or memo that you can add to the withdrawal ticket.
ExternalAddress	string . The place to route withdrawn funds.

Response

The response to **CreateWithdrawTicket** is a standard response object confirming receipt or non-receipt of the information, and is not the ticket *per se*. To view and confirm the ticket information, use the call **GetWithdrawTicket**.

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

Where:

String	Value
result	Boolean . If the call has been successfully received by the Order Management System, result is <i>true</i> ; otherwise, it is <i>false</i> .
errmsg	string . A successful receipt of the call returns null; the <i>errmsg</i> 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 <i>errorcodes</i> shown in the <i>errmsg</i> list.
detail	string . Message text that the system may send. The content of this parameter is usually <i>null</i> .

See Also

ConfirmWithdraw, **GetWithdraws**, **GetWithdrawTemplate**,
GetWithdrawTemplateTypes, **GetWithdrawTicket**, **UpdateWithdrawTicket**.

GetAllDepositRequestInfoTemplates

Certain account providers require specific information (deposit request info) for deposits. For example, a deposit via credit card might require the credit card number, first, and last name. Other payment providers will require other sets of information.

A call to **GetAllDepositRequestInfoTemplates** returns an array of template objects applicable to the product (asset).

Request

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

Where:

String	Value
OMSId	integer. The ID of the Order Management System.
ProductId	integer. The ID of the product (asset) that determines the template information.

Response

The response is an array of available template objects and response information that confirms whether the call was successful. Within each template is the *depositTemplate* object and within that, a JSON string (called *Template*) that holds specific data in string/value pairs. Two *Templates* objects are returned in this example response.

```
{
  "Templates": [
    {
      "providerId": 10,
      "providerName": "Nigerian Naira",
      "depositTemplate": {
        "ProviderType": "QuickTeller",
        "Template": "{
          \"FirstName\": null,
          \"LastName\": null,
          \"Amount\": 0
        }",
        "ProcessInfo": " ",
        "UseGetDepositWorkflow": true,
        "DepositWorkflow": "CryptoWallet"
      }
    },
    {
      "providerId": 7,
      "providerName": "Nigerian Naira",
      "depositTemplate": {
        "ProviderType": "WebPay",
        "Template": "{
          \"FirstName\": null,
          \"LastName\": null,
          \"Amount\" :null
        }"
      }
    }
  ]
}
```

GetAllDepositRequestInfoTemplates

Code continued from page 199

```
    },
    "ProcessInfo": " ",
    "UseGetDepositWorkflow": true,
    "DepositWorkflow": "MerchantRedirect"
  }
],
"result": false,
"errormsg": null,
"statuscode": 0
}"
}
```

Where:

String	Value
Templates	Array of Objects. See <i>Templates</i> object, following.
result	Boolean. Returns <i>true</i> if the call has been successfully received by the Order Management System; otherwise <i>false</i> .
errormsg	string. A successful receipt of the call returns null; the <i>errormsg</i> parameter for an unsuccessful call returns one of the following strings: NotAuthorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
statuscode	integer. A value for <i>statuscode</i> is returned by this call, but is not used.

Templates object. The code sample shows an example array of *Templates* objects. They vary according to the asset being deposited and the account provider.

String	Value
providerId	integer. The ID of the account provider, an external third party that handles funds and crypto-currencies being deposited and withdrawn.
providerName	string. The name of the account provider.
depositTemplate	object. An account provider-specific set of string/value pairs that hold deposit information. The <i>depositTemplate</i> includes a JSON string object called <i>Template</i> that may hold additional information such as name and amount of the deposit.
ProcessInfo	string. The <i>ProcessInfo</i> string also varies with the provider and the asset being deposited. In a generic deposit template, this string/value pair will be <i>empty</i> ; in other cases it will be an address for processing the deposit.
UseGetDepositWorkflow	Boolean. A <i>true</i> value causes the deposit to use <i>DepositWorkflow</i> ; a <i>false</i> value causes the deposit not to use <i>DepositWorkflow</i> .

Table continued from page 200

DepositWorkflow	string. A set of enumerated values that describe the deposit workflow for this template. One of: CryptoWallet ManualDeposit MerchantForm MerchantRedirect Custom
-----------------	--

Template JSON object. Within the *depositTemplate* object is the Template JSON stringify object. In this example case, it holds the name of the depositor and the amount being deposited. Other templates may hold other information.

String	Value
FirstName	string. First name of the depositor.
LastName	string. Last name of the depositor.
Amount	real. Amount being deposited.

See Also

CreateDepositTicket, GetAllDepositTickets, GetDepositInfo,
GetDepositRequestInfoTemplate, GetDepositTicket, UpdateDepositTicket

GetAllDepositTickets

Returns all deposit tickets that match the string/value pairs included in the request, starting at a specific ticket, and returning up to a total number that can be specified in the request.

Note: Only admin-level users can issue this call.

Request

OMSId and *OperatorId* are required; other string/value pairs are optional. *AmountOperator* must be included if an *Amount* value is included.

```
{
  "OMSId": 1,
  "OperatorId": 1,
  "AccountId": 1,
  "Status": 5,
  "TicketId": 127,
  "StartTimestamp": "2016-04-21T21:48:22Z",
  "EndTimestamp": "2016-11-21T21:48:22Z",
  "StartIndex": 0,
  "Limit": 0,
  "UserName": "John Smith",
  "Amount": 50,
  "AmountOperator": 1,
}
```

Where:

String	Value
OMSId	integer. The ID of the Order Management System on which the deposit tickets reside. Required.
OperatorId	integer. The ID of the operator of the trading venue. Required.
AccountId	integer. The ID of the account
Status	integer. The current status of the deposit. One of: 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending Note: The value of <i>Status</i> is an integer in the request for GetAllDepositTickets . In the response, it is a string..
TicketId	integer. The ID of a single deposit ticket that is unique across the Order Management System. By including a value for <i>TicketId</i> , you limit the returned information to a single ticket.

GetAllDepositTickets

Table continued from page 203

StartIndex	integer. Optional. The deposit ticket at which to start returning the array of deposit tickets, starting from 0, the most recent deposit ticket, and working backwards in time.
Limit	integer. Optional. The total number of deposit tickets to return in the array. <i>Limit</i> is a 32-bit integer value that can return over 4 billion tickets (4 thousand million). If <i>Limit</i> is not specified, all tickets are returned.
UserName	string. The name of the user making the deposit.
StartTimestamp	string. The start of the period over which to return deposit tickets, in ISO 8601 format. See "Time— and Date-Stamp Formats" on page 8.
EndTimestamp	string. The end of the period over which to return deposit tickets, in ISO 8601 format. See "Time— and Date-Stamp Formats" on page 8.
Amount	real. The amount of the deposit. If you specify an <i>Amount</i> value, you must include <i>AmountOperator</i> .
AmountOperator	integer. Tells the response to return tickets in ranges based on the <i>Amount</i> value. This string/value pair is required if you specify an <i>Amount</i> value. There is no <i>AmountOperator</i> setting for greater-than or less-than an <i>Amount</i> value. 0 returns tickets with values equal to the <i>Amount</i> value. 1 returns tickets with values equal to or greater than the <i>Amount</i> value. 2 returns tickets with values less than or equal to the <i>Amount</i> value.

Response

The response is an array of deposit ticket information, with each element in the array representing one ticket. The number of elements in the returned array is set by the value of the *Limit* string in the request.

```
[
  {
    "AssetManagerId": 1,
    "AccountId": 1,
    "AssetId": 1,
    "AssetName": "BTC",
    "Amount": 100.0,
    "OMSID": 1,
    "RequestCode": "6D2E6447-AED7-4E5B-8759-B2F564E95FC7",
    "RequestIP": "90.171.32.77",
    "RequestUser": 1,
    "RequestUserName": "admin",
    "OperatorId": 1,
    "Status": "New",
    "FeeAmt": 0.0,
    "UpdatedByUser": 1,
    "UpdatedByUserName": "admin",
    "TicketNumber": 127,
    "DepositInfo": "{
      \"Full Name\": \"John Smith\",
      \"language\": \"en\",
      \"Bank Name\": \"\"
    }",
    "CreatedTimestamp": "2016-04-21T21:48:22Z",
    "LastUpdateTimeStamp": "2016-04-21T21:48:22Z",
    "Comments": [],
    "Attachments": [],
  },
]
```

Where:

String	Value
AssetManagerId	Integer. The ID of the Asset Manager module, which interacts with the OMS and the trading venue's matching engine. The Asset Manager accepts, holds, and disburses assets (products). See "Modules" on page 3 for more background on the Asset Manager.
AccountId	integer. The ID of the account into which the deposit was made.
AssetId	Integer. The ID of the asset being deposited. Equivalent to product ID. <i>AssetId</i> = <i>ProductId</i> , and uses the same ID numbers.
AssetName	string. The name of the asset being deposited. USD (dollars), BTC (bitcoin), gold, NZD (New Zealand dollars) for example. This is not an enumerated field, to allow for flexibility.
Amount	real. The amount of the asset being deposited.
OMSId	integer. The ID of the Order Management System handling the deposits.
RequestCode	string. A GUID (globally unique ID) string that identifies this specific deposit.
RequestIp	string. The on-line IP (Internet Protocol) address from which the deposit is made. This can be a traditional IPv4 dotted quad (192.168.168.1) or a 128-bit IPv6 address.
RequestUser	integer. The ID of the user sending the request and making the deposit.
RequestUserName	string. The name of the user sending the request and making the deposit. For example, "John Smith."
OperatorId	integer. The ID of the operator of the trading venue.
Status	string. The current status of the deposit. One of: 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending Note: The value of <i>Status</i> is an integer in the request for GetAllDepositTickets . In the response, it is a string..
FeeAmt	real. The value of the fee for making the deposit, if any.
UpdatedByUser	integer. The ID of the most recent user updating this deposit ticket.
UpdateByUserName	string. The name of the most recent user updating this deposit ticket, for example, "Joan Smith."
TicketNumber	integer. A system-assigned unique deposit ticket number that identifies the deposit. The value for <i>TicketNumber</i> is returned by the GetDepositTicket~ calls: GetAllDepositTickets and GetDepositTicket ,

GetAllDepositTickets

Table continued from page 205

DepositInfo	string. A list of strings and string/value pairs that holds information about the source of funds being deposited. This information was entered when the deposit ticket was created, and as required by the account provider.
CreatedTimestamp	string. The time and date that the deposit was created, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.
LastUpdateTimeStamp	string. The time and date that the deposit ticket last was updated, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8.
Comments	array of strings. <i>Comments</i> are sets of system-generated string/value pairs that provide information about the deposit’s process through the system. Neither users nor admins enter these comments directly.
Attachments	array of strings. A set of base-64 strings usually providing an image or a PDF. This image or file may be a transaction receipt or other information that the depositor wishes to attach to the deposit for record-keeping purposes.

See Also

CreateDepositTicket, GetDepositInfo, GetDepositTicket, UpdateDepositTicket.

GetAllWithdrawTickets

Returns *all* withdraw tickets from the specified Order Management System and trading venue operator, without limitation. This is an administrator-level call.

Note: Only Admin-level users can issue this call.

Request

OMSId and *OperatorId* are required; other string/value pairs are optional. *AmountOperator* must be included if an *Amount* value is included.

```
"OMSId": 1,  
"OperatorId": 1,  
"AccountId": 1,  
"Status": 5,  
"TicketId": 127,  
"StartTimestamp": "2016-04-21T21:48:22Z",  
"EndTimestamp": "2016-11-21T21:48:22Z",  
"StartIndex": 0,  
"Limit": 0,  
"UserName": "John Smith",  
"Amount": 50,  
"AmountOperator": 1,
```

Where:

String	Value
OMSId	integer. The ID of the Order Management System on which the withdraw tickets reside. Required.
OperatorId	integer. The ID of the operator of the trading venue. Required.
AccountId	integer. The ID of the account
Status	integer. The current status of the withdrawal. One of: 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending 8 Pending2Fa 9 AutoAccepted 10 Delayed Note: The value of <i>Status</i> is an integer in the request for GetAllWithdrawTickets . In the response, it is a string..
TicketId	integer. The ID of a single withdraw ticket that is unique across the Order Management System. By including a value for <i>TicketId</i> , you limit the returned information to a single ticket.

GetAllWithdrawTickets

Table continued from page 207

StartIndex	integer. Optional. The withdraw ticket at which to start returning the array of withdraw tickets, starting from 0, the most recent ticket, and working backwards in time.
Limit	integer. Optional. The total number of withdraw tickets to return in the array. <i>Limit</i> is a 32-bit integer value that can return over 4 billion tickets (4 thousand million). If <i>Limit</i> is not specified, all tickets are returned.
UserName	string. The name of the user making the withdrawal.
StartTimestamp	string. The start of the period over which to return withdraw tickets, in ISO 8601 format. See “Time— and Date-Stamp Formats” on page 8.
EndTimestamp	string. The end of the period over which to return withdraw tickets, in ISO 8601 format. See “Time— and Date-Stamp Formats” on page 8.
Amount	real. The amount of the withdrawal. If you specify an <i>Amount</i> value, you must include <i>AmountOperator</i> .
AmountOperator	integer. Tells the response to return tickets in ranges based on the <i>Amount</i> value. This string/value pair is required if you specify an <i>Amount</i> value. There is no <i>AmountOperator</i> setting for greater-than or less-than an <i>Amount</i> value. 0 returns tickets with values equal to the <i>Amount</i> value. 1 returns tickets with values equal to or greater than the <i>Amount</i> value. 2 returns tickets with values less than or equal to the <i>Amount</i> value.

Response

The response is an array of withdraw ticket objects and comments.

```
[
  {
    "AssetManagerId": 0,
    "AccountId": 4,
    "AssetId": 2,
    "AssetName": "Tether",
    "Amount": 0.4,
    "TemplateForm": "{
      \"TemplateType\": \"TetherRPCWithdraw\",
      \"Comment\": \"TestWithdraw\",
      \"ExternalAddress\": \"ms6C3pKAAr8gRCcnVebs8VRkVrjcvqNYv3\",
    }",
    "TemplateFormType": "TetherRPCWithdraw",
    "OMSId": 1,
    "RequestCode": "490b4fa3-53fc-44f4-bd29-7e16be86fba3",
    "RequestIP": "108.35.121.205",
    "RequestUserId": 7,
    "RequestUserName": "testUser6",
    "OperatorId": 1,
    "Status": "FullyProcessed",
    "FeeAmt": 0,
    "UpdatedByUser": 7,
    "UpdatedByUserName": "testUser6",
    "TicketNumber": 51,
    "CreatedTimestamp": "2017-11-22T20:15:54Z",
    "LastUpdateTimestamp": "2017-11-22T20:16:11Z",
    "Comments": [
      {
        "CommentId": 2,
        "EnteredBy": 1,
        "EnteredDateTime": "2017-11-22T20:16:11Z",
        "Comment": "Withdraw Submitted (11/22/2017 8:16:11 PM): 2c843665606bdfdcdbdf9a307a89098f5b68e53448287e33b672d3f1090bd49a7",
      }
    ]
  }
]
```

Code continued on page 209

Code continued from page 208

```

        "OperatorId": 1,
        "OMSId": 1,
        "TicketCode": "49541cc89-c9c3-4bf6-9141-da1f0ef67fae",
        "TicketId": 51
    },
    ],
    "Attachments": [],
    "AuditLog": []
},
...]
```

Where:

String	Value
AssetManagerId	integer. The ID of the Asset Manager module. See "Modules" on page 3 for more information about the Asset Manager.
AccountId	integer. The ID of the account that made the withdrawal.
AssetId	integer. The ID of the asset in which the withdrawal is denominated, for example, US Dollar or BitCoin both have an associated <i>AssetId</i> . <i>AssetId</i> and <i>ProductId</i> are identical in numerical content. You must use <i>AssetId</i> here.
AssetName	string. The readable name of the asset in which the withdrawal is denominated, for example, "US Dollar" or "BitCoin."
Amount	real. The amount of the withdrawal.
TemplateForm	object. See the <i>TemplateForm</i> object, following. The content of a template depends on the account provider that you use for deposits and withdrawals. This template is provided as a general reference example. See "Deposit and Withdraw Templates" on page 9 for more information about templates.
TemplateFormType	string. The name of the template being used. The template controls the string/value pairs in the <i>TemplateForm</i> object returned for each withdrawal. These vary by account provider. See "Deposit and Withdraw Templates" on page 9 for more information about templates.
OMSId	integer. The ID of the Order Management System.
RequestCode	string. A GUID (globally unique ID) string that identifies this specific withdrawal.
RequestIP	string. The IP address from which the withdrawal was initiated, in either IPv4 dotted quad format or IPv6 format.
RequestUserId	integer. The ID of the user who made the original withdrawal request.
RequestUserName	string. The name of the user who made the original withdrawal request.
OperatorId	integer. The ID of the operator of the trading venue on which the withdrawal request was made.

Table continued on page 210

GetAllWithdrawTickets

Table continued from page 209

Status	<p>integer. The current status of the deposit, stated as an integer. One of:</p> <ul style="list-style-type: none"> 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending 8 Pending2Fa 9 AutoAccepted 10 Delayed <hr/> <p>Note: Withdraw tickets include <i>Status</i> values 8 through 10, which do not apply to deposit tickets. <i>Status</i> for GetAllWithdrawTickets and GetAllDepositTickets are numerical; other instances of <i>Status</i> are strings.</p>
FeeAmt	<p>real. The amount of any fee that was charged for the withdrawal. <i>FeeAmt</i> is always denominated in the asset or product of the withdrawal, for example in US Dollars, BitCoin, or other currency, depending on the nature of the funds being withdrawn.</p>
UpdatedByUser	<p>integer. The ID of any user who made an update to the withdraw ticket. Updates are most usually to <i>Status</i>.</p>
UpdatedByUsername	<p>string. The name of any user who made an update to the withdraw ticket. Updates are most usually to <i>Status</i>.</p>
TicketNumber	<p>integer. A system-assigned unique withdraw ticket number that identifies the withdrawal. The value for <i>TicketNumber</i> is returned by the Get~ calls: GetAllWithdrawTickets and GetWithdrawTicket.</p>
CreatedTimestamp	<p>string. The time and date at which the withdraw ticket was created, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8 for more information about this format.</p>
LastUpdateTimestamp	<p>string. If the ticket has been updated, shows the time and date stamp of the update in ISO 8601 format; if the ticket has not been updated, shows the same time and date stamp as CreateTimestamp. See “Time– and Date-Stamp Formats” on page 8 for more information on ISO 8601.</p>
Comments	<p>array of strings. <i>Comments</i> are sets of system-generated string/value pairs that provide information about the withdraw ticket’s process through the system. Neither users nor admins enter these comments directly.</p>
Attachments	<p>array of strings. A set of base-64 strings usually providing an image or a PDF. This image or file may be a transaction receipt or other information that the depositor wishes to attach to the deposit for record-keeping purposes.</p>
AuditLog	<p>array. Reserved for future use.</p>

TemplateForm object. The content of a template depends on the needs of the account provider that the account uses for deposits and withdrawals. This template information is provided as a general reference to show what a template looks like. See “Deposit and Withdraw Templates” on page 9 for more information about templates. This example comes from the *TemplateFormType* “TetherRpcWithdraw.”

String	Value
TemplateType	string. The name of the template.
Comment	string. Any comment pertaining to the withdrawal.
ExternalAddress	string. An external address supplied by the account provider to accept the withdrawal.

Sample *Comment* string/value pairs

The *Comment* string shown in the code sample may be one of an array of comments that track a withdrawal through the system and across updates. The string/value pairs are created by the system. Neither users nor admins can enter comments directly.

String	Value
CommentId	integer. An arbitrary ID identifying this comment in the array of comments. Best practice: assign such an ID sequentially.
EnteredBy	integer. The ID of the user entering the comment.
EnteredDateTime	string. The date and time that the comment was entered. This example shows ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8 for more information on the time-and-date-stamp formats supported.
Comment	string. The body of the comment. The example comment shows the date-and-time-stamp when the withdrawal was submitted, and a cryptographic code.
OperatorId	integer. The ID of the operator of the trading venue.
OMSId	integer. The ID of the Order Management System from which the withdrawal was made. Note that this <i>OMSId</i> may or may not be the same as the <i>OMSId</i> string/value pair in the request and response of the enclosing GetAllWithdrawTickets call.
TicketCode	string. A GUID (globally unique ID) that matches the <i>RequestCode</i> value in the response. The <i>RequestCode</i> uniquely identifies the withdrawal.
TicketId	integer. The system assigns the comment-level <i>TicketId</i> . It is associated with the ticket’s <i>TicketNumber</i> value.

See Also

ConfirmWithdraw, CreateWithdrawTicket, GetWithdrawTemplate,
GetWithdrawTemplateTypes, GetWithdrawTicket,
GetWithdrawTickets, UpdateWithdrawTicket

GetDepositInfo

Returns an array of cryptographic deposit addresses that the user will use to send crypto-currency funds to the trading venue.

Note: The keys returned by **GetDepositInfo** are only for crypto-currency deposits.

Request

The request always returns a set of cryptographic keys that each act as an address for sending coins from a crypto-currency wallet. *GenerateNewKey* generates a new set of cryptographic keys the software suggests a new key set with each deposit.

```
{
  "OMSIId": 1,
  "AccountId": 4,
  "ProductId": 1,
  "GenerateNewKey": true
}
```

Where:

String	Value
OMSIId	integer. The ID of the Order Management System into which the deposits were made.
AccountId	integer. The ID of the account into which the deposits were made.
ProductId	integer. The ID of the product or asset in which the deposits were made. Product ID and Asset ID are equivalent. See "Products and Instruments" on page 4 for more information about products and instruments.
GenerateNewKey	Boolean. A <i>true</i> value means that the system will generate a new (never before seen) cryptographic key for use with deposits. The new key will be returned as the first element of the set of keys. A value of <i>false</i> provides the set of keys previously used. The software suggests using a new key set with each deposit.

Response

The response returns a set of cryptographic keys for use in making deposits from a crypto-currency wallet, along with any error message and confirmation of asset manager module, account, account provider, and asset (product). Select one of the keys for use with making the transfer from the crypto-currency wallet.

```
{
  "AssetManagerId": 1,
  "AccountId": 4,
  "AssetId": 1,
  "ProviderId": 1,
  "DepositInfo": "[
    "mkqpdfHecor7QCd53sEyWE1TL4UupE8L9z",
    "muTjY641lMmRTccvyHxJmET3t3b8EeCNT",
    "mjN4bJJotcCRRiiGpWXYiE4vdRpVMQf5Wa",
    "mivkpkPBKm5sXFGiySgVbhxQ2Keb9fKGiR",
  ]"
```

Code continued on page 214

GetDepositInfo

Code continued from page 213

```
    "mvAKe9ZoNLWALYuc2TK9HVK9dt5oLsbAAZ",
    "myCP69kfd3eulMvW8DKXjffsEJns2mK3K5",
    "mubnHGgBqBunjGSsx25bbBA4n2eBpJGJKC",
    "mxV5wizeS2r6j8jiUWUsTrMimda3fSgx8f",
    "mr5Llqb4cJTsvJPnU4zMD95snXdNsJpd6L",
    "mqt2NgUrhYQWixcNdU76u5uAa3u34tqvmr",
    "mgySiizawpSMeJhz3Q7K1w9zgEwE4JQ4fD"
  ],
  "result": true,
  "errmsg": null,
  "statusCode": 0
}
```

Where:

String	Value
AssetManagerId	integer. The ID of the asset manager module for this installation. See “Modules” on page 3 for more information about the Order Management System, matching engine, and Asset Manager modules.
AccountId	integer. The ID of the account into which the deposit will be made (and for which the set of cryptographic keys will be valid).
AssetId	integer. The ID of the asset in which the deposit is denominated, for example, US Dollars or BitCoin. <i>AssetId</i> is numerically equivalent to <i>ProductId</i> .
ProviderId	integer. The ID of the third-party Account Provider who makes the conversion between national currency and crypto-currency.
DepositInfo	string. The response returns a set of cryptographic keys for use in making deposits from a crypto-currency wallet. We suggest generating new keys for each new deposit. Note: Do not confuse this <i>DepositInfo</i> , which returns cryptographic keys, with the <i>DepositInfo</i> string/value pairs created as part of a deposit ticket or returned by GetDepositTicket or GetAllDepositTickets .
result	Boolean. Returns <i>true</i> if the request for deposit info was received correctly; returns <i>false</i> if the request was not received correctly.
errmsg	string. A successful receipt of the call returns null; the <i>errmsg</i> 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)
statusCode	integer. If <i>result</i> is <i>false</i> , <i>statusCode</i> can return: 32 Not_Authorized 33 AssetManager_Not_Found If no account provider is located, <i>statusCode</i> returns <i>null</i> .

See Also

CreateDepositTicket, GetAllDepositTickets, GetDepositTicket, UpdateDepositTicket.

GetDepositRequestInfoTemplate

This call returns the single most appropriate deposit template available to you for a given set of product (asset), account, and account provider (the third-party organization that handles your deposits and withdrawals). In this case, the template provides the specific information that an account provider requires to make a deposit.

Request

```
{
  "OMSIId": 1,
  "ProductId": 8,
  "AccountId": 4,
  "AccountProviderId": 10
}
```

Where:

String	Value
OMSIId	integer. The ID of the Order Management System.
ProductId	integer. The ID of the product or asset that determines the template information.
AccountId	integer. The ID of the account that returns the deposit request info.
AccountProviderId	integer. <i>Optional.</i> The ID of the account provider that handles the deposit. You must include this if there are multiple account providers.

Response

The response returns the deposit template appropriate to the set of product (asset) being deposited, the account into which the deposit is being made, and the account provider. The template shown here is just an example. There is a wide variety of templates and information they require.

```
{
  "Template": {
    "ProviderType": "BitcoinRpc",
    "Template": "{}",
    "ProcessInfo": "",
    "UseGetDepositWorkflow": true,
    "DepositWorkflow": "CryptoWallet"
  },
  "result": true,
  "errmsg": null,
  "statusCode": 0
}
```

GetDepositRequestInfoTemplate

Where:

String	Value
Template	Object. See <i>Template</i> object, following.
result	Boolean. Returns <i>true</i> if the call has been successfully received by the Order Management System; otherwise <i>false</i> .
errormsg	string. A successful receipt of the call returns null; the <i>errormsg</i> parameter for an unsuccessful call returns one of the following strings: NotAuthorized (errorcode 20) Invalid Request (errorcode 100) Operation Failed (errorcode 101) Server Error (errorcode 102) Resource Not Found (errorcode 104)
statuscode	integer. A value for <i>statuscode</i> is returned by this call, but is not used.

Template object. The *Template* object returns a set of string/value pairs that contain information for the deposit; among those string/value pairs is a JSON string object (also called *Template*) that may contain additional string/values pairs required by the account provider. This example does not include any — the brackets are empty — but they may include items such as first name and last name of the depositor.

String	Value
ProviderType	string. The type of asset handled by the account provider; in this example, BitCoin.
Template	JSON object. An account provider-specific set of string/value pairs that will vary from account provider to account provider.
ProcessInfo	string. The <i>ProcessInfo</i> string also varies with the provider and the asset being deposited. In a generic deposit template, the <i>ProcessInfo</i> string/value pair will be <i>empty</i> ; in other cases it will be an address for processing the deposit.
UseGetDepositWorkflow	Boolean. A <i>true</i> value causes the deposit to use <i>DepositWorkflow</i> ; a <i>false</i> value causes the deposit not to use <i>DepositWorkflow</i> .
DepositWorkflow	string. A set of enumerated values that describe the deposit workflow for this template. One of: CryptoWallet ManualDeposit MerchantForm MerchantRedirect Custom

See Also

CreateDepositTicket, GetAllDepositTickets, GetAllDepositRequestInfoTemplates, GetDepositInfo, GetDepositTicket, UpdateDepositTicket

GetDepositTicket

Returns a single deposit ticket by matching its request code to one already in the database.

Note: Only admin-level users can issue this call.

Request

You can find the *RequestCode* value for a ticket by calling **GetAllDepositTickets** or by retaining the *RequestCode* value returned when the ticket was created.

```
{
  "OMSIId": 1,
  "OperatorId": 1,
  "RequestCode": "866f21fe-3461-41d1-91aa-5689bc38503f",
  "AccountId": 4
}
```

Where:

String	Value
OMSIId	integer. The ID of the Order Management System.
OperatorId	integer. The ID of the operator of the trading venue.
RequestCode	string. A GUID (globally unique ID) string that identifies this specific deposit. <i>RequestCode</i> is part of the response returned to CreateDepositTicket .
AccountId	integer. The ID of the account into which the deposit was made.

Response

The response returns information about the single deposit ticket that matches the *RequestCode* in the request.

```
{
  "AssetManagerId": 1,
  "AccountId": 4,
  "AssetId": 2,
  "AssetName": "US Dollar",
  "Amount": 10,
  "OMSIId": 1,
  "RequestCode": "866f21fe-3461-41d1-91aa-5689bc38503f",
  "RequestIP": "90.171.32.77",
  "RequestUser": 1,
  "RequestUserName": "admin",
  "OperatorId": 1,
  "Status": "New",
  "FeeAmt": 0,
  "UpdatedByUser": 1,
  "UpdatedByUserName": "admin",
  "TicketNumber": 67,
  "DepositInfo": "{
    \"Full Name\": \"Test\",
    \"language\": \"en\",
  }
```

Code continued on page 218

GetDepositTicket

Code continued from page 217

```
"Bank Name": ""
},
"CreatedTimestamp": "2017-12-14T15:13:31Z",
"LastUpdateTimeStamp": "2017-12-14T15:13:31Z",
"Comments": [],
"Attachments": []
}
```

Where:

String	Value
AssetManagerId	integer. The ID of the Asset Manager module. See "Modules" on page 3 for more information about the Asset Manager.
AccountId	integer. The ID of the account into which this deposit was made.
AssetId	integer. The ID of the asset in which the deposit is denominated. <i>AssetId</i> and <i>ProductId</i> are identical in numerical content.
AssetName	string. The name of the asset — for example, "US Dollar."
Amount	real. The amount of the deposit.
OMSId	integer. The ID of the Order Management System where the account resides that received the deposit.
RequestCode	string. A GUID (globally unique identifier) and identifies this and only this deposit. This string/value pair echoes and confirms the value sent in the request.
RequestIP	string. The IP address (in IPv4 dotted quad notation or IPv6 notation) from which the original deposit was made.
RequestUser	integer. The ID of the user who made the deposit.
RequestUserName	string. The name of the user who made the deposit.
OperatorId	integer. The ID of the operator of the trading venue on which the deposit was made.
Status	string. The current status of the deposit, stated as an string. One of: 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending Note: <i>Status</i> is an integer in GetAllDepositTickets and in GetAllWithdrawTickets . In GetDepositTicket and all other calls that use <i>Status</i> , it is a string.
FeeAmt	real. The amount of any fee charged for the deposit. <i>FeeAmt</i> always is denominated in the asset being deposited.
UpdatedByUser	integer. If this deposit ticket has been updated, this string/value pair shows the ID of the user who updated it. See UpdateDepositTicket .

Table continued on page 219

Table continued from page 218

UpdatedByUserName	string. If this deposit ticket has been updated, this string/value pair shows the name of the user who updated it.
TicketNumber	integer. A system-assigned unique deposit ticket number that identifies the deposit. The value for <i>TicketNumber</i> is returned by the GetDepositTicket~ calls: GetAllDepositTickets and GetDepositTicket .
DepositInfo	string. See <i>DepositInfo</i> object, below. <i>DepositInfo</i> holds string/value pairs with information about the deposit.
CreateTimestamp	string. The time and date that the deposit ticket was created, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8 for more information.
LastUpdateTimeStamp	string. The time and date that the deposit ticket was last updated, in ISO 8601 format. “Time– and Date-Stamp Formats” on page 8. If the deposit ticket was not updated, this string/value pair holds the same value as <i>CreateTimestamp</i> .
Comments	array of strings. <i>Comments</i> are sets of system-generated string/value pairs that provide information about the deposit’s process through the system. Neither users nor admins enter these comments directly.
Attachments	array of strings. A set of base-64 strings usually providing an image or a PDF. This image or file may be a transaction receipt or other information that the depositor wishes to attach to the deposit for record-keeping purposes.

DepositInfo object

String	Value
Full Name	string. The name of the user who made the deposit.
language	string. A two-letter abbreviation for the language in which the deposit is written — <i>en</i> for English, <i>sp</i> for Spanish, <i>ru</i> for Russian, and so forth. These abbreviations are not enumerated to allow for flexibility.
Bank Name	string. The name of the bank from which the deposit was made.

See Also

CreateDepositTicket, GetAllDepositTickets, GetDepositInfo, UpdateDepositTicket.

GetWithdrawTemplate

Returns the text of the withdraw template that you name. You can obtain the name of an asset- and account-provider-appropriate template by using **GetWithdrawTemplateTypes**.

Request

```
{
  "OMSIId": 1,
  "AccountId": 4,
  "ProductId": 2},
  "templateType": "Standard"
}
```

Where:

String	Value
OMSIId	integer . The ID of the Order Management System on which the withdrawal will be made.
AccountId	integer . The ID of the account from which the withdrawal will be made.
ProductId	integer . The ID of the product (asset) that will be withdrawn. <i>AssetId</i> and <i>ProductId</i> are numerically equivalent. You must use <i>ProductId</i> here.
templateType	string . The name of the withdraw template whose text you want to return.

Response

The response returns the string/value pairs of the template, along with any error messages about the call.

```
{
  "Template": "{
    \"FullName\": null,
    \"Language\": null,
    \"Comment\": null,
    \"BankAddress\": null,
    \"BankAccountNumber\": null,
    \"BankAccountName\": null,
    \"SwiftCode\": null}",
  "result": true,
  "errorMsg": null,
  "statusCode": 0
}
```

GetWithdrawTemplate

Where:

String	Value
Template	object. The text of the template that the call returns will vary with the account, asset (product), and account provider handling the withdrawal. See the example Template object, following.
result	Boolean. If the call has been successfully received by the Order Management System, the result is <i>true</i> ; otherwise, it is <i>false</i> .
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)
statuscode	integer. If <i>result</i> is <i>false</i> , <i>statuscode</i> can return: 32 Not_Authorized 33 AssetManager_Not_Found If no account provider is located, <i>statuscode</i> returns <i>null</i> .

Template Object. A template object is a string containing a set of string/value pairs. The template returned by **GetWithdrawTemplate** will vary with the account, asset (product), and account provider. This example is taken from the “Standard” template.

String	Value
FullName	string. Name of the person making the withdrawal.
Lanugage	string. A two-character abbreviation of the language of the withdrawal information, for example, En for English, Sp for Spanish, or Ru for Russian. The values for this string are not enumerated, to allow for variation and expansion.
Comment	string. A comment entered by the withdrawer.
BankAddress	string. The physical address of the bank or account provider — 123 Fourth St., for example.
BankAccountNumber	string. The account at the bank or other account provider that received the funds withdrawn from the OMS and trading venue account.
BankAccountName	string. The name associated with the account at the bank or other account provider that will receive the funds withdrawn from the OMS and trading venue account.
SwiftCode	string. An international bank code that uniquely identifies a specific bank internationally. It is also known as a Bank Identifier Code, or BIC. The Swift Code consists of 8 or 11 alphanumeric characters.

See Also

GetWithdrawTemplateTypes

GetWithdrawTemplateTypes

Returns an array of names of withdrawal template that are appropriate to the asset (product) you want to withdraw from the account. Obtain the withdrawal template itself by calling **GetWithdrawTemplate** with the correct name.

Request

```
{
  "OMSId": 1,
  "AccountId": 4,
  "ProductId": 2
}
```

Where:

String	Value
OMSId	integer. The ID of the Order Management System.
AccountId	integer. The ID of the account from which you want to make the withdrawal.
ProductId	integer. The ID of the product (asset) that you want to withdraw. <i>ProductId</i> and <i>AssetId</i> are identical in value, but this call uses <i>ProductId</i> .

Response

The response returns an array of template names, and a success code.

```
{
  "TemplateTypes": [
    "Standard"
  ],
  "result": true,
  "errmsg": null,
  "statusCode": 0
}
```

Where:

String	Value
TemplateTypes	array of strings. The templates that are appropriate to the withdrawal of your asset, account, and Order Management System. If the call was unsuccessful, <i>TemplateTypes</i> may return an empty array.
result	Boolean. If the call has been successfully received by the Order Management System, result is <i>true</i> ; otherwise, it's <i>false</i> .

GetWithdrawTemplateTypes

Table continued from page 223

errormsg	string. A successful receipt of the call returns null; the <i>errormsg</i> 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)
statuscode	integer. If <i>result</i> is <i>false</i> , <i>statuscode</i> can return: 32 Not_Authorized 33 AssetManager_Not_Found If no account provider is located, <i>statuscode</i> returns <i>null</i> ..

See Also

[GetWithdrawTemplate](#)

GetWithdrawTicket

Returns a single withdraw ticket from the Order Management System, trading venue operator, and account that matches the GUID (globally unique identifier) in *RequestCode*. Obtain the GUID from the call **CreateWithdrawTicket** when the ticket is first created, or from **GetAllWithdrawTickets**, another admin-level-only call. An administrator can use **GetWithdrawTicket** to return any single withdrawal ticket in the system.

Note: Only admin-level users can use this call.

Request

```
{
  "OMSId": 1,
  "OperatorId": 1,
  "RequestCode": "2ddcbcd6-74c5-4082-8e62-ee93862a2c2d",
  "AccountId": 4
}
```

Where:

String	Value
OMSId	integer. The ID of the Order Management System where the withdrawal was made.
OperatorId	integer. The ID of the trading venue operator on the system where the withdraw was made.
RequestCode	string. A GUID (globally unique ID) that identifies the specific withdrawal ticket you want to return. Obtain the RequestCode from CreateWithdrawTicket or GetAllWithdrawTickets .
AccountId	integer. The ID of the account from which the withdrawal was made.

Response

The response returns information about a single withdraw ticket.

```
{
  "AssetManagerId": 0,
  "AccountId": 4,
  "AssetId": 2,
  "AssetName": "US Dollar",
  "Amount": 10,
  "TemplateForm": "{
    \"FullName\": \"TestUser\",
    \"Language\": \"En\",
    \"Comment\": \"Test comment\",
    \"BankAddress\": \"My Bank's address\",
    \"BankAccountNumber\": \"123456789\",
    \"BankAccountName\": \"MyBank\",
    \"SwiftCode\": \"SWIFT1023\"
  }",
  "TemplateFormType": "Standard",
}
```

Code continued on page 226

GetWithdrawTicket

Code continued from page 225

```
{
  "OMSIId": 1,
  "RequestCode": "2ddcbcd6-74c5-4082-8e62-ee93862a2c2d",
  "RequestIP": "90.171.32.77",
  "RequestUserId": 1,
  "RequestUserName": "admin",
  "OperatorId": 1,
  "Status": "Pending2Fa",
  "FeeAmt": 0,
  "UpdatedByUser": 1,
  "UpdatedByUserName": "admin",
  "TicketNumber": 52,
  "CreatedTimestamp": "2017-12-14T15:25:13Z",
  "LastUpdateTimestamp": "2017-12-14T15:25:13Z",
  "Comments": [],
  "Attachments": [],
  "AuditLog": []
}
```

Where:

String	Value
AssetManagerId	integer. The ID of the Asset Manager module from which the withdrawal was made. See "Modules" on page 3 for more information about system components.
AccountId	integer. The ID of the account from which the withdrawal was made.
AssetId	integer. The ID of the asset in which the withdrawal was denominated, for example in US Dollars or BitCoin. <i>AssetId</i> has the same value content as <i>ProductId</i> .
AssetName	string. The name of the asset in which the withdrawal was denominated, for example, "US Dollars" or "BitCoin."
Amount	real. The amount of the withdrawal.
TemplateForm	object. See the <i>TemplateForm</i> object, following. The content of a template depends on the account provider that you use for deposits and withdrawals. This template is provided only as a general reference example. See "Deposit and Withdraw Templates" on page 9 for more information about templates.
TemplateFormType	string. The name of the template being used. The template controls the string/value pairs in the <i>TemplateForm</i> object returned for each withdrawal. These vary by account provider. See "Deposit and Withdraw Templates" on page 9 for more information about templates.
OMSIId	integer. The ID of the Order Management System from which the withdrawal was made. See "Modules" on page 3 for more information about system components.
RequestCode	string. A GUID (globally unique ID) that identifies the specific withdraw ticket.
RequestIP	string. The IP address (either in IPv4 dotted quad format or in IPv6) from which the original withdraw ticket was submitted.
RequestUserId	integer. The ID of the user who originally submitted the withdraw request.
RequestUserName	string. The name of the user who originally submitted the withdraw request.
OperatorId	integer. The ID of the operator of the trading venue.

Table continued from page 226

Status	<p>string. The current status of the withdrawal, stated as a string. One of:</p> <ul style="list-style-type: none"> 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending 8 Pending2FA 9 AutoAccepted 10 Delayed <hr/> <p>Note: Withdrawal calls have three additional <i>Status</i> values that deposit calls do not: <i>Status</i> values 8 through 10.</p>
UpdatedByUser	integer. If the withdraw ticket has been updated, this string/value pair shows the ID of the account that updated the ticket.
UpdatedByUsername	string. If the withdraw ticket has been updated, this string/value pair shows the name of the person who updated the ticket.
TicketNumber	integer. A system-assigned unique withdraw ticket number that identifies the withdrawal. The value for <i>TicketNumber</i> is returned by the Get~ calls: GetAllWithdrawTickets and GetWithdrawTicket ,
CreatedTimestamp	string. The date and time that the withdraw ticket was first created, in ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8 for more information on these formats.
LastUpdateTimeStamp	string. The date and time that the withdraw ticket was last updated. If the withdraw ticket has not been updated, this string/value pair will show the same time as <i>CreatedTimeStamp</i> . See “Time– and Date-Stamp Formats” on page 8.
Comments	array of strings. <i>Comments</i> are sets of system-generated string/value pairs that provide information about the withdrawal’s process through the system. Neither users nor admins can enter these comments directly.
Attachments	array of strings. A set of base-64 strings usually providing an image or a PDF. This image or file may be a transaction receipt or other information that the depositor wishes to attach to the deposit for record-keeping purposes.
AuditLog	array of strings. Reserved for future use.

TemplateForm object. The *TemplateForm* varies from account provider to the needs of the account provider. This is only an example of one template.

String	Value
FullName	string. The name of the bank account holder, as it appears on the bank account.
Language	string. A two-character abbreviation of the language of the withdrawal information, for example, En for English, Sp for Spanish, or Ru for Russian. The values for this string are not enumerated, to allow for variation and expansion.
Comment	string. Any comment that was applied to the withdraw ticket.

Table continued on page 228

GetWithdrawTicket

Table continued from page 227

BankAddress	string. The physical address of the bank or other account provider that received the funds withdrawn from the OMS and trading venue account.
BankAccountNumber	string. The account at the bank or other account provider that received the funds withdrawn from the OMS and trading venue account.
BankAccountName	string. The name associated with the account at the bank or other account provider that received the funds withdrawn from the OMS and trading venue account.
SwiftCode	string. An international bank code that uniquely identifies a specific bank internationally. It is also known as a Bank Identifier Code, or BIC. The Swift Code consists of 8 or 11 alphanumeric characters.

See Also

[ConfirmWithdraw](#), [CreateWithdrawTicket](#), [GetWithdrawTemplate](#),
[GetWithdrawTemplateTypes](#), [GetWithdrawTickets](#), [UpdateWithdrawTicket](#)

GetWithdrawTickets

Returns an array of withdraw tickets from the specified Order Management System and Account.

Note: Any level user can issue **GetWithdrawTickets**. **GetAllWithdrawTickets** and **GetWithdrawTicket** are admin-only calls.

Request

```
{
  "OMSId": 1,
  "AccountId": 4,
  "StartIndex": 0,
  "Limit": 200
}
```

Where:

String	Value
OMSId	integer. The ID of the Order Management System from which withdrawals were made.
AccountId	integer. The ID of the account from which withdrawals were made.
StartIndex	integer. Optional. The withdraw ticket at which to start returning the array of withdraw tickets, starting from 0, the most recent withdraw ticket, and working backwards in time.
Limit	integer. Optional. The total number of withdraw tickets to return in the array. <i>Limit</i> is a 32-bit integer value that can return over 4 billion tickets (4 thousand million). If <i>Limit</i> is not specified, all tickets are returned.

Response

The response is an array of withdraw ticket objects, template information, and comments.

```
[
  {
    "AssetManagerId": 0,
    "AccountId": 4,
    "AssetId": 2,
    "AssetName": "US Dollar",
    "Amount": 12,
    "TemplateForm": "{
      \"FullName\": \"John Smith\",
      \"Language\": \"en\",
      \"Comment\": \"Withdrawal 10/18/18\",
      \"BankAddress\": \"mybank.com\",
      \"BankAccountNumber\": \"12345A27\",
      \"BankAccountName\": \"John Smith & Sons\",
      \"SwiftCode\": \"ABCDUS27\"
    }",
    "TemplateFormType": "Standard",
    "OMSId": 1,
    "RequestCode": "490b4fa3-53fc-44f4-bd29-7e16be86fba3",
  }
]
```

Code continued on page 230

GetWithdrawTickets

Code continued from page 229

```

"RequestIP": "99.127.45.221",
"RequestUserId": 1,
"RequestUserName": "admin",
"OperatorId": 1,
"Status": "AdminCancelled",
"FeeAmt": 0,
"UpdatedByUser": 1,
"UpdatedByUserName": "admin",
"TicketNumber": 49,
"CreatedTimestamp": "2017-11-17T17:05:52Z",
"LastUpdateTimestamp": "2017-11-17T17:11:31Z",
"Comments": [
  {
    "CommentId": 28,
    "EnteredBy": 1,
    "EnteredDateTime": "2017-11-17T17:08:13Z",
    "Comment": "Withdraw Submitted (11/17/2017 5:08:13 PM): ",
    "OperatorId": 1,
    "OMSId": 1,
    "TicketCode": "4fa1674c-bf29-4a37-a5ef-0197dc9fc5d4",
    "TicketId": 48
  },
  {
    "CommentId": 27,
    "EnteredBy": 1,
    "EnteredDateTime": "2017-11-17T17:05:53Z",
    "Comment": "Withdraw Failed: Exceeds_Daily-Withdraw_Limit",
    "OperatorId": 1,
    "OMSId": 1,
    "TicketCode": "490b4fa3-53fc-44f4-bd29-7e16be86fba3",
    "TicketId": 49
  }
],
"Attachments": [],
"AuditLog": []
}],
...]
```

Where:

String	Value
AssetManagerId	integer. The ID of the Asset Manager module. See “Modules” on page 3 for more information about the Asset Manager.
AccountId	integer. The ID of the account from which the withdrawal was made.
AssetId	integer. The ID of the asset in which the withdrawal is denominated, for example, US Dollar or BitCoin both have an associated <i>AssetId</i> . <i>AssetId</i> and <i>ProductId</i> are identical in numerical content. You must use <i>AssetId</i> here.
AssetName	string. The readable name of the asset in which the withdrawal is denominated, for example, “US Dollar” or “BitCoin.”
Amount	real. The amount of the withdrawal.
TemplateForm	object. See the <i>TemplateForm</i> object, following. The content of a template depends on the account provider that you use for deposits and withdrawals. This template is provided as a general reference. See “Deposit and Withdraw Templates” on page 9 for more information about templates.

Table continued from page 230

TemplateFormType	string. The name of the template being used. The template controls the string/value pairs in the <i>TemplateForm</i> object returned for each withdrawal. These vary by account provider. See "Deposit and Withdraw Templates" on page 9 for more information about templates.
OMSId	integer. The ID of the Order Management System.
RequestCode	string. A GUID (globally unique ID) string that identifies this specific withdrawal.
RequestIP	string. The IP address from which the withdrawal was initiated, in either IPv4 dotted quad format or IPv6 format.
RequestUserId	integer. The ID of the user who made the original withdrawal request.
RequestUserName	string. The name of the user who made the original withdrawal request.
OperatorId	integer. The ID of the operator of the trading venue on which the withdrawal request was made.
Status	<p>string. The current status of the deposit, stated as a string. One of:</p> <ul style="list-style-type: none"> 0 New 1 AdminProcessing 2 Accepted 3 Rejected 4 SystemProcessing 5 FullyProcessed 6 Failed 7 Pending 8 Pending2Fa 9 AutoAccepted 10 Delayed <hr/> <p>Note: Withdraw tickets include <i>Status</i> values 8 through 10, which do not apply to deposit tickets.</p>
FeeAmt	real. The amount of any fee that was charged for the withdrawal. <i>FeeAmt</i> is always denominated in the asset or product of the withdrawal, for example in US Dollars, BitCoin, or other currency, depending on the nature of the funds being withdrawn.
UpdatedByUser	integer. The ID of the most recent user who made an update to the withdraw ticket. Updates are most usually made for <i>Status</i> changes.
UpdatedByUserName	string. The name of the most recent user who made an update to the withdraw ticket. Updates are most usually made for <i>Status</i> changes.
TicketNumber	integer. A system-assigned unique withdraw ticket number that identifies the withdrawal. The value for <i>TicketNumber</i> is returned by the Get~ calls: GetAllWithdrawTickets and GetWithdrawTicket ,
CreatedTimestamp	string. The time and date at which the withdraw ticket was created, in ISO 8601 format. See "Time- and Date-Stamp Formats" on page 8 for more information about this format.
LastUpdateTimestamp	string. If the ticket has been updated, shows the time and date stamp of the update in ISO 8601 format; if the ticket has not been updated, shows the same time and date stamp as CreateTimestamp. See "Time- and Date-Stamp Formats" on page 8 for more information on ISO 8601.

Table continued on page 232

GetWithdrawTickets

Table continued from page 231

Comments	array of strings. Comments are sets of system-generated string/value pairs that provide information about the withdrawal's process through the system. Neither users nor admins enter these comments directly. See "Sample <i>Comment</i> string/value pairs" on page 232 for an explanation of the <i>Comments</i> section used in this code example.
Attachments	array of strings. A set of base-64 strings usually providing an image or a PDF. This image or file may be a transaction receipt or other information that the depositor wishes to attach to the deposit for record-keeping purposes.
AuditLog	array. Reserved for future use.

TemplateForm object. The content of a template depends on the needs of the account provider that the account uses for deposits and withdrawals. This template information is provided as a general reference to show what a template looks like. See "Deposit and Withdraw Templates" on page 9 for more information about templates. This example comes from the *TemplateFormType* "Standard."

String	Value
FullName	string. The name of the bank account holder, as it appears on the bank account.
Language	string. A two-character abbreviation of the language of the withdrawal information, for example, En for English, Sp for Spanish, or Ru for Russian. The values for this string are not enumerated, to allow for variation and expansion.
Comment	string. Any comment that was applied to the withdraw ticket.
BankAddress	string. The physical (street) address of the bank or other account provider that received the funds withdrawn from the OMS and trading venue account.
BankAccountNumber	string. The account at the bank or other account provider that received the funds withdrawn from the OMS and trading venue account.
BankAccountName	string. The name associated with the account at the bank or other account provider that received the funds withdrawn from the OMS and trading venue account. For example, "John Smith & Sons."
SwiftCode	string. An international bank code that uniquely identifies a specific bank internationally. It is also known as a Bank Identifier Code, or BIC. The Swift Code consists of 8 or 11 alphanumeric characters.

Sample *Comment* string/value pairs

The *Comment* string shown in the code sample may be one of an array of comments that track a withdrawal through the system and across updates. The string/value pairs are created by the system. Neither users nor admins can enter comments directly. These are example pairs only.

String	Value
CommentId	integer. An arbitrary ID identifying this comment in the array of comments.
EnteredBy	integer. The ID of the user entering the comment.

Table continued from page 232

EnteredDateTime	string. The date and time that the comment was entered. This example shows ISO 8601 format. See “Time– and Date-Stamp Formats” on page 8 for more information on the time-and-date-stamp formats supported.
Comment	string. The body of the comment.
OperatorId	integer. The ID of the operator of the trading venue.
OMSId	integer. The ID of the Order Management System from which the withdrawal was made. Note that this <i>OMSId</i> may or may not be the same as the <i>OMSId</i> string/value pair in the request and response of the enclosing GetWithdrawTickets call.
TicketCode	string. A GUID (globally unique ID) that matches the <i>RequestCode</i> value in the response. The <i>RequestCode</i> uniquely identifies the withdrawal.
TicketId	integer. The system assigns the comment-level <i>TicketId</i> . It is associated with the ticket's <i>TicketNumber</i> value.

See Also

ConfirmWithdraw, CreateWithdrawTicket, GetAllWithdrawTickets, GetWithdrawTemplate, GetWithdrawTemplateTypes, GetWithdrawTicket, UpdateWithdrawTicket

Index

A

Aborted

request status 10

Aborting

request status 10

about exchange software 1

account

events, subscribing to 167

Account. *See also* AccountId

GetOpenOrders 76
GetOpenQuotes 82
GetOrderHistoryByOrderId 93
GetOrdersHistory 97
GetOrderStatus 102

AccountHandle

GetAccountInfo 53, 54
GetUserAccountInfos 112

accountId

GenerateTradeActivityReport 129

AccountId

CancelAllOrders 41
CancelOrder 43
CancelQuote 45
CancelReplaceOrder 48
CreateDepositTicket 193
CreateQuote 51
CreateWithdrawTicket 197
GetAccountInfo 53, 54
GetAccountPositions 57, 58
GetAccountTrades 59, 60
GetAccountTransactions 63, 64
GetAllDepositTickets 203, 205
GetAllWithdrawTickets 207, 209
GetDepositInfo 213, 214
GetDepositRequestInfoTemplate 215
GetDepositTicket 217, 218
GetOpenOrders 75
GetOpenQuotes 79
GetOrderFee 85
GetOrderHistory 87
GetOrdersHistory 95
GetOrderStatus 101
GetTradesHistory 164, 165
GetUserAccountInfos 112
GetUserInfo 29
GetWithdrawTemplate 221
GetWithdrawTemplateTypes 223
GetWithdrawTicket 225, 226

GetWithdrawTickets 229, 230
SendOrder 117 SetUserInfo 37, 38
SubscribeAccountEvents 167
UpdateDepositTicket 235
UpdateQuote 121
UpdateWithdrawTicket 239

accountIdList

GenerateTradeActivityReport 127
GenerateTransactionActivityReport 131
GenerateTreasuryActivityReport 135

AccountIdList

ScheduleTradeActivityReport 145
ScheduleTransactionActivityReport 149
ScheduleTreasuryActivityReport 153

accountIds

GenerateTransactionActivityReport 133
GenerateTreasuryActivityReport 137
GetUserReportTickets 141
ScheduleTradeActivityReport 147
ScheduleTransactionActivityReport 151
ScheduleTreasuryActivityReport 155

account information 53, 111

AccountName

GetAccountInfo 54
GetUserAccountInfos 112

AccountPositionEvent

SubscribeAccountEvents 168 trigger event: account balance changes 168

AccountProviderId

GetDepositRequestInfoTemplate 215

accounts

multiple users of 4

Accounts

SubscribeLevel2 160, 176

account transactions 63

AccountType

GetAccountInfo 54
GetUserAccountInfos 112

ActionDateTime

SubscribeLevel2, 160

ActionType

SubscribeLevel2 160, 176

AffiliatedId

GetUserInfo 30
RegisterNewUser 13
SetUserInfo 38

Amount

CreateDepositTicket 193
CreateWithdrawTicket 197
GetAccountPositions 58
GetAllDepositRequestInfoTemplates 201
GetAllDepositTickets 204, 205
GetAllWithdrawTickets 208, 209
GetDepositTicket 218
GetOrderFee 85
GetWithdrawTicket 226
GetWithdrawTickets 230
UpdateDepositTicket 235
UpdateWithdrawTicket 240

AmountOperator

GetAllDepositTickets 204
GetAllWithdrawTickets 208

APEX Extract CSV Data

Dictionary 127, 131, 135

API calls

contents common to many calls 7
quote-related and order-related 5
wrap in message frame 1

ask 161

Ask

CreateQuote 51
GetOpenQuotes 81
UpdateQuote 121

AskQty

CreateQuote 51
UpdateQuote 121

AskQuoteId

CancelQuote 45
CreateQuote 52
UpdateQuote 121, 122

Ask quotes

GetOpenQuotes 79

AskResult

CancelQuote 46
CreateQuote 52
UpdateQuote 122

Asset

same as Product 4

Index

AssetId

- CreateDepositTicket 193
- equal to ProductId 205
- GetAllDepositTickets 205
- GetAllWithdrawTickets 209
- GetDepositInfo 214
- GetDepositTicket 218
- GetWithdrawTicket 226
- GetWithdrawTickets 230
- UpdateDepositTicket 235
- UpdateWithdrawTicket 239

Asset Manager 4

AssetManagerId

- GetAllDepositTickets 205
- GetAllWithdrawTickets 209
- GetDepositInfo 214
- GetDepositTicket 218
- GetWithdrawTicket 226
- GetWithdrawTickets 230
- UpdateDepositTicket 235
- UpdateWithdrawTicket 239

AssetName

- CreateDepositTicket 193
- GetAllDepositTickets 205
- GetAllWithdrawTickets 209
- GetDepositTicket 218
- GetWithdrawTicket 226
- GetWithdrawTickets 230
- UpdateDepositTicket 235
- UpdateWithdrawTicket 240

Attachments

- GetAllDepositTickets 206
- GetAllWithdrawTickets 210
- GetDepositTicket 219
- GetWithdrawTicket 227
- GetWithdrawTickets 232
- UpdateDepositTicket 236
- UpdateWithdrawTicket 241

AuditLog

- GetAllWithdrawTickets 210
- GetWithdrawTicket 227
- GetWithdrawTickets 232
- UpdateWithdrawTicket 241

Authenticate2FA 17

- Authenticated 17
- authenticated calls 3
- Code 17
- SessionToken 17

Authenticated

- Authenticate2FA 17
- WebAuthenticateUser 15

authenticated and public calls 3

authenticated calls

- Authenticate2FA 3
- GetInstrument 3
- GetInstruments 3
- GetProduct 3
- GetProducts 3

- GetTickerHistory 3
- LogOut 3
- SubscribeLevel1 3
- SubscribeLevel2 3
- SubscribeTicker 3
- UnsubscribeLevel1 3
- UnsubscribeLevel2 3
- UnsubscribeTicker 3
- WebAuthenticateUser 3

authentication

- two-factor 17

AvgPrice

- GetOpenOrders 77
- GetOpenQuotes 82
- GetOrderHistory 89
- GetOrderHistoryByOrderId 93
- GetOrdersHistory 98
- GetOrderStatus 103

B

background information 1

Balance

- GetAccountTransactions 64

BankAccountName

- GetWithdrawTemplate 222
- GetWithdrawTicket 228
- GetWithdrawTickets 232
- TemplateForm 228, 232

BankAccountNumber

- GetWithdrawTemplate 222
- GetWithdrawTicket 228
- GetWithdrawTickets 232
- TemplateForm 228, 232

BankAddress

- GetWithdrawTemplate 222
- GetWithdrawTicket 228
- GetWithdrawTickets 232
- TemplateForm 228, 232

Bank Name

- CreateDepositTicket 194
- GetDepositTicket 219

beginTime

- ScheduleTradeActivityReport 145
- ScheduleTransactionActivityReport 149
- ScheduleTreasuryActivityReport 153

BestBid

- SubscribeLevel1 172

BestOffer

- SubscribeLevel1 172

bid 161

Bid

- CreateQuote 51

- GetOpenQuotes 81
- UpdateQuote 121

BidQty

- CreateQuote 51
- UpdateQuote 121

BidQuoteId

- CancelQuote 45
- CreateQuote 52
- UpdateQuote 121, 122

Bid quotes

- GetOpenQuotes 79

BidResult

- CancelQuote 46
- CreateQuote 52
- UpdateQuote 122

BlockTrade

- SubscribeTrades 181
- types of orders 7

C

calls

- authenticated and public 3

CancelAllOrders 41

- AccountId 41
- detail 3, 42
- errorcode 3, 42
- errmsg 3, 42
- InstrumentId 42
- OMSId 41
- result 2, 42
- UserId 41

CancelAllOrdersRejectEvent

- SubscribeAccountEvents 168 trigger event: all orders rejected 168

CancelOrder 43

- AccountId 43
- ClientOrderId 43
- detail 44
- errorcode 44
- errmsg 44
- OMSId 43
- OrderId 43
- result 44

CancelOrderRejectEvent

- SubscribeAccountEvents 168 trigger event: order is canceled 168

CancelQuote 45

- AccountId 45
- AskQuoteId 45
- AskResult 46
- BidQuoteId 45
- BidResult 46
- detail 46
- errorcode 46

- errormsg 46
- InstrumentId 45
- OMSId 45
- result 46
- CancelReplaceOrder 47.** *See also ModifyOrder*
 - AccountId 48
 - ClientOrderId 48
 - InstrumentId 48
 - LimitPrice 48
 - OMSId 48
 - OrderIdOCO 49
 - OrderIdToReplace 48
 - OrderType 48
 - OrigClOrdId 49
 - OrigOrderId 49
 - PegPriceType 48
 - Quantity 49
 - ReplacementClOrdId 49
 - ReplacementOrderId 49
 - Side 48
 - StopPrice 48
 - TimeInForce 49
 - TrailingAmount 48
- CancelReplaceOrderRejectEvent**
 - SubscribeAccountEvents 169
 - trigger event: order rejected even with cancel-replace 169
- CancelUserReport 125**
 - detail 126
 - errorcode 125
 - errormsg 125
 - GUID 125
 - result 125
 - UserReport 125
- ChangeReason**
 - GetOpenOrders 77
 - GetOpenQuotes 83
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
 - GetOrderStatus 103
- ClientOrderId**
 - CancelOrder 43
 - CancelReplaceOrder 48
 - GetOpenOrders 76
 - GetOpenQuotes 82
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 95, 98
 - GetTradesHistory 165
 - SendOrder 117
- ClientOrderID**
 - GetOrderStatus 103
- close 161**
- Code**
 - Authenticate2fa 17
- Comment**
 - CreateWithdrawTicket 198
 - example template 211
 - GetAllWithdraTickets 211
 - GetWithdrawTemplate 222
 - GetWithdrawTicket 227
 - GetWithdrawTickets 232, 233
 - sample comments 211, 233
 - Templateform 227, 232
 - UpdateWithdrawTicket 241
- CommentId**
 - GetAllWithdraTickets 211
 - GetWithdrawTickets 232
 - sample comment 211, 232
- Comments**
 - GetAllDepositTickets 206
 - GetAllWithdrawTickets 210
 - GetDepositTicket 219
 - GetWithdrawTicket 227
 - GetWithdrawTickets 232
 - UpdateDepositTicket 236
 - UpdateWithdrawTicket 241
- comment, sample 211, 232**
- common error codes 2**
- Completed**
 - request status 10
- compliance information**
 - storing 27
- Config**
 - SetUserConfig 35
- Config strings**
 - adding 35
 - deleting 33
- contents common to many API calls 7**
- Count**
 - GetAccountTrades 59
- Counterparty**
 - GetAccountTransactions 64
- CounterParty**
 - GetAccountTrades 61
 - GetTradesHistory 166
- CounterPartyId**
 - GetOpenOrders 77
 - GetOpenQuotes 82
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
 - GetOrderStatus 103
- CR**
 - GetAccountTransactions 64
- create an order 117**
- CreateDepositTicket 193**
 - AccountId 193
 - Amount 193
- AssetId 193
- AssetName 193
- Bank Name 194
- DepositInfo 194
- deposits of fiat money 193
- Full Name 194
- language 194
- OMSId 193
- OperatorId 193
- requestcode 194
- RequestUser 193
- Status 194
- success 194
- CreatedTimestamp**
 - GetAllDepositTickets 206
 - GetAllWithdrawTickets 210
 - GetWithdrawTicket 227
 - GetWithdrawTickets 231
 - UpdateDepositTicket 236
 - UpdateWithdrawTicket 241
- CreateQuote 51**
 - AccountId 51
 - Ask 51
 - AskQty 51
 - AskQuoteId 52
 - AskResult 52
 - Bid 51
 - BidQty 51
 - BidQuoteId 52
 - BidResult 52
 - detail 52
 - errorcode 52
 - errormsg 52
 - InstrumentId 51
 - OMSId 51
 - result 52
- createTime**
 - GenerateTradeActivityReport 128
 - GenerateTransactionActivityReport 132
 - GenerateTreasuryActivityReport 136
 - GetUserReportTickets 140
 - ScheduleTradeActivityReport 147
 - ScheduleTransactionActivityReport 151
 - ScheduleTreasuryActivityReport 155
- CreateTimestamp**
 - GetDepositTicket 219
- create user 13**
- CreateWithdrawTicket 197**
 - AccountId 197
 - Amount 197
 - Comment 198
 - detail 198
 - errorcode 198
 - errormsg 198
 - ExternalAddress 198
 - OMSId 197
 - ProductId 197
 - result 198

Index

- TemplateForm 197
- TemplateType 197, 198
- credit entry for account 64**
- crypto-currency**
 - deposits 213
- cryptographic keys for deposit use 213**
- CryptoWallet**
 - template 216
- CurrentDayNumTrades**
 - SubscribeLevel1 172
- CurrentDayPxChange**
 - SubscribeLevel1 172
- CurrentDayVolume**
 - SubscribeLevel1 172
- Custom**
 - template 216
- custom information**
 - adding 35
 - deleting 33
- custom key/value pairs 27**

D

- data fee**
 - ticker 177
- DateTimeCreated**
 - GetUserInfo 30
 - SetUserInfo 38
- debit entry for account 64**
- DecimalPlaces**
 - GetProduct 106, 108
- DepositInfo**
 - CreateDepositTicket 194
 - GetAllDepositTickets 206
 - GetDepositInfo 214
 - GetDepositTicket 219
 - UpdateDepositTicket 236
- deposits**
 - crypto-currency 213
 - obtaining cryptographic keys 213
 - subscribing to 167
- deposits of fiat money**
 - CreateDepositTicket 193
- deposits, tracked by system**
 - non-crypto currencies 193
- Deposits, tracked by system**
 - crypto-currencies 213
- deposit template**
 - GetDepositRequestInfoTemplate 215

- depositTemplate**
 - GetAllDepositRequestInfoTemplates 200
- deposit templates**
 - GetAllDepositRequestInfoTemplates 199
- deposit ticket**
 - finding 217
 - update 235
- deposit tickets**
 - finding 203
- DepositWorkflow**
 - GetAllDepositRequestInfoTemplates 201
 - GetDepositRequestInfoTemplate 216
- Depth**
 - GetAccountTransactions 63
 - GetL2Snapshot 159
 - GetOrdersHistory 96
 - GetTradesHistory 164
 - SubscribeLevel2 175
 - used as count of trades 63, 95, 163
 - used as depth of market 159, 175
- descriptorId**
 - GetUserReportWriterResultRecords 143
- detail**
 - CancelAllOrders 42
 - CancelOrder 44
 - CancelQuote 46
 - CancelUserReport 126
 - CreateQuote 52
 - CreateWithdrawTicket 198
 - in generic response object 3, 198
 - LogOut 19
 - ModifyOrder 116
 - RemoveUserConfig 34
 - SetUserConfig 36
 - UnsubscribeLevel1 183
 - UnsubscribeLevel2 185
 - UnsubscribeTicker 187
 - UnsubscribeTrades 189
 - UpdateDepositTicket 237
 - UpdateQuote 122
 - UpdateWithdrawTicket 242
- details about a product 105**
- Direction**
 - GetAccountTrades 61
 - GetTradesHistory 166
 - SubscribeTrades 180
- Display Quantity**
 - defined 8
- DisplayQuantity**
 - GetOpenOrders 76
 - GetOpenQuotes 82
 - GetOrderHistory 88
 - GetOrderHistoryByOrderId 92

- GetOrdersHistory 97
- GetOrderStatus 102
- SendOrder 117

DR

- GetAccountTransactions 64

E

Email

- GetUserInfo 29
- RegisterNewUser 13
- SetUserInfo 37, 38

EmailVerified

- GetUserInfo 29
- SetUserInfo 37, 38

endTime

- GenerateTradeActivityReport 127
- GenerateTransactionActivityReport 131
- GenerateTreasuryActivityReport 135

EndTimestamp

- GetAllDepositTickets 204
- GetAllWithdrawTickets 208
- GetOrdersHistory 96

EndTimeStamp

- GetTradesHistory 164

EnteredBy

- GetAllWithdraTickets 211
- GetOpenOrders 77
- GetOpenQuotes 83
- GetOrderHistory 89
- GetOrderHistoryByOrderId 93
- GetOrdersHistory 98
- GetOrderStatus 103
- GetWithdrawTickets 232
- sample comment 211, 232

EnteredDateTime

- GetAllWithdraTickets 211
- GetWithdrawTickets 233
- sample comment 211, 233

enter user information

- SetUserInfo 37

errorcode

- CancelAllOrders 42
- CancelOrder 44
- CancelQuote 46
- CancelUserReport 125
- CreateQuote 52
- CreateWithdrawTicket 198
- in generic response object 3
- LogOut 19
- ModifyOrder 116
- RemoveUserConfig 34
- SetUserConfig 36
- UnsubscribeLevel1 183
- UnsubscribeLevel2 185

- UnsubscribeTicker 187
- UnsubscribeTrades 189
- UpdateDepositTicket 237
- UpdateQuote 122
- UpdateWithdrawTicket 242
- errorcode 20**
 - Not Authorized 3, 198, 214, 237
- errorcode 100**
 - Invalid Request 3, 198, 214, 237
- errorcode 101**
 - Operation Failed 3, 198, 214, 237
- errorcode 102**
 - Server Error 3, 198, 214, 237
- errorcode 104**
 - Resource Not Found 3, 198, 214, 237
- error codes, common 2**
- errmsgs**
 - 20, not authorized 3, 198, 214, 237
 - 100, invalid request 3, 198, 214, 237
 - 101, operation failed 3, 198, 214, 237
 - 102, server error 3, 198, 214, 237
 - 104, resource not found 3, 198, 214, 237
 - CancelAllOrders 42
 - CancelOrder 44
 - CancelQuote 46
 - CancelUserReport 125
 - CreateQuote 52
 - CreateWithdrawTicket 198
 - GetAllDepositRequestInfoTemplates 200
 - GetDepositInfo 214
 - GetDepositRequestInfoTemplate 216
 - GetWithdrawTemplate 222
 - GetWithdrawTemplateTypes 224
 - in generic response object 3
 - LogOut 19
 - ModifyOrder 116
 - RemoveUserConfig 34
 - SendOrder 119
 - SetUserConfig 36
 - UnsubscribeLevel1 183
 - UnsubscribeLevel2 185
 - UnsubscribeTicker 187
 - UnsubscribeTrades 189
 - UpdateDepositTicket 237
 - UpdateQuote 122
 - UpdateWithdrawTicket 242
- estimate fees 85**
- events**
 - AccountPositionEvent 168
 - CancelAllOrdersRejectEvent 168
 - CancelOrderRejectEvent 168
 - CancelReplaceOrderRejectEvent 169
 - MarketStateUpdate 169
 - NewOrderRejectEvent 169
 - OrderStateEvent 169
 - OrderTradeEvent 170

- PendingDepositUpdate 170
- example template 211**
- ExecutionId**
 - GetAccountTrades 60
 - GetTradesHistory 164, 165
- ExternalAddress**
 - CreateWithdrawTicket 198
 - example template 211
 - GetAllWithdrawTickets 211
 - UpdateWithdrawTicket 241

F

- Fee**
 - GetAccountTrades 60
 - GetTradesHistory 165
- FeeAmt**
 - GetAllDepositTickets 205
 - GetAllWithdrawTickets 210
 - GetDepositTicket 218
 - GetWithdrawTickets 231
 - UpdateDepositTicket 236
 - UpdateWithdrawTicket 240
- FeeGroupID**
 - GetAccountInfo 54
 - GetUserAccountInfos 112
- FeeProduct**
 - GetAccountInfo 55
 - GetUserAccountInfos 113
- FeeProductId**
 - GetAccountTrades 60
 - GetTradesHistory 165
- FeeProductType**
 - GetAccountInfo 54
 - GetUserAccountInfos 112
- fiat money**
 - CreateDepositTicket 193
- find a single deposit ticket 217**
- find a single withdraw ticket 225**
- FirmId**
 - GetAccountInfo 54
 - GetUserAccountInfos 112
- FirmName**
 - GetAccountInfo 54
 - GetUserAccountInfos 112
- FirstName**
 - GetAllDepositRequestInfoTemplates 201
- FOK (Fill Or Kill) 49**
- frequency**
 - ScheduleTradeActivityReport 145

- ScheduleTransactionActivityReport 149
- ScheduleTreasuryActivityReport 153

- FromDate**
 - GetTickerHistory 161
- Full Name**
 - CreateDepositTicket 194
 - GetDepositTicket 219
- FullName**
 - GetWithdrawTemplate 222
 - GetWithdrawTicket 227
 - GetWithdrawTickets 232
 - TemplateForm 227, 232

G

- GenerateNewKey**
 - GetDepositInfo 213
- GenerateTradeActivityReport 127**
 - accountId 129
 - accountIdList 127
 - createTime 128
 - endTime 127
 - initialRunTime 128
 - intervalDuration 129
 - intervalEndTime 129
 - intervalStartTime 128
 - lastInstanceId 129
 - omsId 127
 - OMSId 128
 - reportFlavor 128
 - ReportFrequency 129
 - RequestId 129
 - RequestingUser 128
 - requestStatus 129
 - startTime 127
- GenerateTransactionActivityReport 131**
 - accountIdList 131
 - accountIds 133
 - createTime 132
 - endTime 131
 - initialRunTime 132
 - intervalDuration 133
 - intervalEndTime 133
 - intervalStartTime 132
 - lastInstanceId 133
 - omsId 131
 - OMSId 132
 - reportFlavor 132
 - ReportFrequency 133
 - RequestId 133
 - RequestingUser 132
 - requestStatus 133
 - startTime 131

GenerateTreasuryActivityReport 135

- accountIdList 135
- accountIds 137
- createTime 136
- endTime 135
- initialRunTime 136
- intervalDuration 137
- intervalEndTime 137
- intervalStartTime 136
- lastInstanceId 137
- omsId 135
- OMSId 136
- reportFlavor 136
- ReportFrequency 137
- RequestId 137
- RequestingUser 136
- requestStatus 137
- startTime 135

generic response object 2

GetAccountInfo 53

- AccountHandle 53, 54
- AccountId 53, 54
- AccountName 54
- AccountType 54
- FeeGroupID 54
- FeeProduct 55
- FeeProductType 54
- FirmId 54
- FirmName 54
- OMSId 53, 54
- ParentID 54
- RefererId 55
- RiskType 54
- SupportedVenueIds 55
- VerificationLevel 54

GetAccountPositions 57

- AccountId 57, 58
- Amount 58
- Hold 58
- OMSId 57, 58
- PendingDeposits 58
- PendingWithdraws 58
- ProductId 58
- ProductSymbol 58
- TotalDayDeposits 58
- TotalDayWithdraws 58
- TotalMonthWithdraws 58

GetAccountTrades 59

- AccountId 59, 60
- Count 59
- CounterParty 61
- Direction 61
- ExecutionId 60
- Fee 60
- FeeProductId 60
- InstrumentId 61
- IsBlockTrade 61
- OMSId 59, 60
- OrderId 60
- OrderOriginator 60

- OrderTradeRevision 61
- Price 61
- Quantity 61
- RemainingQuantity 61
- Side 61
- StartIndex 59
- SubAccountId 60
- TradeId 60
- TradeTime 61
- TradeTimeMS 60
- Value 61

GetAccountTransactions 63

- AccountId 63, 64
- Balance 64
- Counterparty 64
- CR 64
- Depth 63
- DR 64
- OMSId 63, 64
- ProductId 64
- ReferenceId 64
- ReferenceType 64
- TimeStamp 64
- TransactionId 64
- TransactionType 64

GetAllDepositRequestInfoTemplates 199

- Amount 201
- depositTemplate 200
- DepositWorkflow 201
- errmsg 200
- FirstName 201
- LastName 201
- OMSId 199
- ProcessInfo 200
- ProductId 199
- providerId 200
- providerName 200
- result 200
- statuscode 200
- Templates 200
- UseGetDepositWorkflow 200

GetAllDepositTickets 203

- AccountId 203, 205
- Amount 204, 205
- AmountOperator 204
- AssetId 205
- AssetManagerId 205
- AssetName 205
- Attachments 206
- Comments 206
- CreatedTimestamp 206
- DepositInfo 206
- EndTimeStamp 204
- FeeAmt 205
- LastUpdateTimeStamp 206
- Limit 204
- OMSId 203, 205
- OperatorId 203, 205
- RequestCode 205
- RequestIp 205
- RequestUser 205

- RequestUserName 205
- StartIndex 204
- StartTimestamp 204
- Status 203, 205 TicketId 203
- TicketNumber 205
- UpdateByUserName 205
- UpdatedByUser 205
- UserName 204

GetAllWithdraTickets

- Comment 211
- CommentId 211
- EnteredBy 211
- EnteredDateTime 211
- ExternalAddress 211
- OMSId 211
- OperatorId 211
- TemplateType 211
- TicketCode 211
- TicketId 211

GetAllWithdrawTickets 207, 210

- AccountId 207, 209
- Amount 208, 209
- AmountOperator 208
- AssetId 209
- AssetManagerId 209
- AssetName 209
- Attachments 210
- AuditLog 210
- Comments 210
- CreatedTimestamp 210
- EndTimeStamp 208
- FeeAmt 210
- LastUpdateTimestamp 210
- Limit 208
- OMSId 207, 209
- OperatorId 207, 209
- RequestCode 209
- RequestIP 209
- RequestUserId 209
- RequestUserName 209
- StartIndex 208
- StartTimestamp 208
- Status 207, 210
- TemplateForm 209
- TemplateFormType 209
- TicketId 207
- TicketNumber 210
- UpdatedByUser 210
- UpdatedByUserName 210
- UserName 208

GetAvailablePermissionList 25

GetDepositInfo 213

- AccountId 213, 214
- AssetId 214
- AssetManagerId 214
- DepositInfo 214
- errmsg 214
- GenerateNewKey 213
- OMSId 213 ProductId 213

- ProviderId 214
- result 214
- statusCode 214
- GetDepositRequestInfoTemplate 215**
 - AccountId 215
 - AccountProviderId 215
 - DepositWorkflow 216
 - errmsg 216
 - OMSId 215
 - ProcessInfo 216
 - ProductId 215
 - ProviderType 216
 - result 216
 - statusCode 216
 - Template 216
 - UseGetDepositWorkflow 216
- GetDepositTicket 217**
 - AccountId 217, 218
 - Amount 218 AssetId
 - 218 AssetManagerId
 - 218 AssetName 218
 - Attachments 219
 - Bank Name 219
 - Comments 219
 - CreateTimestamp 219
 - DepositInfo 219
 - FeeAmt 218 Full
 - Name 219
 - language 219
 - LastUpdateTimeStamp 219
 - OMSId 217, 218
 - OperatorId 217, 218
 - RequestCode 217, 218
 - RequestIP 218
 - RequestUser 218
 - RequestUserName 218
 - Status 218
 - TicketNumber 219
 - UpdatedByUser 218
 - UpdatedByUserName 219
- GetInstrument 67**
 - authenticated calls 3
 - InstrumentId 67, 68
 - InstrumentType 68
 - OMSId 67, 68
 - PreviousSessionStatus 68
 - Product1 68
 - Product1Symbol 68
 - Product2 68
 - Product2Symbol 68
 - QuantityIncrement 69
 - SelfTradePrevention 69
 - SessionStatus 68
 - SessionStatusDateTime 69
 - SortIndex 68
 - Symbol 68
 - VenueId 68
 - VenueInstrumentId 68
- GetInstruments 71**
 - authenticated calls 3
 - InstrumentId 72
 - InstrumentType 72
 - OMSId 71, 72
 - PreviousSessionStatus 73
 - Product1 72
 - Product1Symbol 72
 - Product2 72
 - Product2Symbol 72
 - QuantityIncrement 73
 - SelfTradePrevention 73
 - SessionStatus 72
 - SessionStatusDateTime 73
 - Symbol 72
 - VenueId 72
 - VenueInstrumentId 72
- GetL2Snapshot 159**
 - Depth 159
 - InstrumentId 159
 - OMSId 159
- GetOpenOrders 75**
 - Account 76
 - AccountId 75
 - AvgPrice 77
 - ChangeReason 77
 - ClientOrderId 76
 - CounterPartyId 77
 - DisplayQuantity 76
 - EnteredBy 77
 - InsideAsk 77
 - InsideBid 77
 - InsideBidSize 77
 - Instrument 76
 - IsLockedIn 77
 - IsQuote 77
 - LastTradePrice 77
 - insideAskSize 77
 - OMSId 75, 77
 - OrderId 76
 - OrderState 77
 - OrderType 76
 - OrigOrderId 77
 - OrigQuantity 77
 - Price 76
 - Quantity 76
 - QuantityExecuted 77
 - ReceiveTime 77
 - ReceiveTimeTicks 77
 - RejectReason 77
 - Side 76
- GetOpenQuotes 79**
 - Account 82
 - AccountId 79
 - Ask 81
 - AvgPrice 82
 - Bid 81
 - Bid and Ask quotes 79
 - ChangeReason 83
 - ClientOrderId 82
 - CounterPartyId 82
 - DisplayQuantity 82
- EnteredBy 83
- InsideAsk 83
- InsideAskSize 83
- InsideBid 83
- InsideBidSize 83
- Instrument 82
- InstrumentId 79
- IsLockedIn 83
- IsQuote 83
- LastTradePrice 83
- OMSId 79, 83
- OrderId 82
- OrderState 82
- OrderType 82
- OrigClOrdId 83
- OrigOrderId 83
- OrigQuantity 82
- Price 82
- Quantity 82
- QuantityExecuted 82
- ReceiveTime 82
- ReceiveTimeTicks 82
- RejectReason 83
- Side 81
- GetOrderFee 85**
 - AccountId 85
 - Amount 85
 - estimate of fee 85
 - InstrumentId 85
 - MakerTaker 86
 - OMSId 85
 - OrderFee 86
 - OrderType 86
 - Price 85
 - ProductId 85, 86
- GetOrderHistory 87**
 - AccountId 87
 - AvgPrice 89
 - ChangeReason 89
 - ClientOrderId 89
 - CounterPartyId 89
 - DisplayQuantity 88
 - EnteredBy 89
 - InsideAsk 90
 - InsideAskSize 90
 - InsideBid 90
 - InsideBidSize 90
 - Instrument 88
 - IsLockedIn 90
 - IsQuote 90
 - LastTradePrice 90
 - OMSId 87, 90
 - OrderId 88
 - OrderState 89
 - OrderType 89
 - OrigClOrdId 89
 - OrigOrderId 89
 - OrigQuantity 89
 - Price 88
 - Quantity 88
 - QuantityExecuted 89
 - ReceiveTime 89

Index

- ReceiveTimeTicks 89
- RejectReason 90 Side 88
- GetOrderHistoryByOrderId 91**
 - Account 93
 - AvgPrice 93
 - ChangeReason 93
 - ClientOrderId 93
 - CounterPartyId 93
 - DisplayQuantity 92
 - EnteredBy 93
 - IndisideBidSize 94
 - InsideAsk 94
 - InsideAskSize 94
 - InsideBid 94
 - Instrument 92
 - IsLockedIn 94
 - IsQuote 94
 - LastTradePrice 94
 - OMSId 91, 94
 - OrderId 91, 92
 - OrderState 93
 - OrderType 93
 - OrigClOrdId 93
 - OrigOrderId 93
 - OrigQuantity 93
 - Price 92
 - Quantity 92
 - QuantityExecuted 93
 - ReceiveTime 93
 - ReceiveTimeTicks 93
 - RejectReason 94
 - Side 92
- GetOrdersHistory 95**
 - Account 97
 - AccountId 95
 - AvgPrice 98
 - ChangeReason 98
 - ClientOrderId 95, 98
 - CounterPartyId 98
 - Depth 96
 - DisplayQuantity 97
 - EndTimestamp 96
 - EnteredBy 98
 - IndisideBidSize 99
 - InsideAsk 98
 - InsideAskSize 98
 - InsideBid 98
 - Instrument 97
 - InstrumentId 95
 - IsLockedIn 99
 - IsQuote 98
 - LastTradePrice 99
 - OMSId 95, 99
 - OrderId 97
 - OrderType 97
 - OrigClOrdId 98
 - OriginalOrderId 95
 - OrigOrderId 98
 - OrigQuantity 98
 - Price 97
 - Quantity 97
- QuantityExecuted 98
- ReceiveTime 98
- ReceiveTimeTicks 98
- RejectReason 99
- Side 97
- StartIndex 96
- StartTimestamp 96
- UserId 95
- GetOrderStatus 101**
 - Account. *See also*
 - AccountId AccountId 101
 - AvgPrice 103
 - ChangeReason 103
 - ClientOrderId 103
 - CounterPartyId 103
 - DisplayQuantity 102
 - EnteredBy 103
 - InsideAsk 104
 - InsideAskSize 104
 - InsideBid 104
 - InsideBidSize 104
 - Instrument. *See also* InstrumentId
 - IsLockedIn 104
 - IsQuote 104
 - LastTradePrice 104
 - OMSId 101, 104
 - OrderId 101, 102
 - OrderState 103
 - OrderType 103
 - OrigClOrdId 103
 - OrigOrderId 103
 - OrigQuantity 103
 - Price 102
 - Quantity 102
 - QuantityExecuted 103
 - ReceiveTime 103
 - ReceiveTimeTicks 103
 - RejectReason 104
 - Side 102
- GetProduct 105**
 - authenticated calls 3
 - DecimalPlaces 106, 108
 - NoFees 106, 108 OMSId 105, 107 Product 106, 108
 - ProductFullName 106, 108
 - ProductId 105, 107
 - ProductType 106, 108
 - TickSize 106, 108
- GetProducts 107**
 - authenticated calls 3
- GetTickerHistory 161**
 - authenticated calls 3
 - FromDate 161
 - InstrumentId 161
- GetTradesHistory 163**
 - AccountId 164, 165
 - ClientOrderId 165
 - CounterParty 166
 - Depth 164
- Direction 166
- EndTimeStamp 164
- ExecutionId 164, 165
- Fee 165
- FeeProductId 165
- InstrumentId 164, 165
- IsBlockTrade 166
- OMSId 163, 165
- OrderId 164, 165
- OrderOriginator 165
- OrderTradeRevision 166
- Price 166
- Quantity 165
- RemainingQuantity 165
- Side 165
- StartIndex 164
- StartTimeStamp 164
- SubAccountId 165
- TradeId 164, 165
- TradeTime 166
- TradeTimeMS 165
- UserId 164
- Value 166
- GetUserAccountInfos 111**
 - AccountHandle 112
 - AccountId 112
 - AccountName 112
 - AccountType 112
 - FeeGroupID 112
 - FeeProduct 113
 - FeeProductType 112
 - FirmId 112
 - FirmName 112
 - OMSId 111
 - OMSID 112
 - ParentId 112
 - RefererId 113
 - RiskType 112
 - SupportedVenueIds 113
 - UserId 111
 - UserName 111
 - VerificationLevel 112
- GetUserAccounts 109**
 - OMSId 109
 - UserId 109
 - UserName 109
- GetUserConfig 27**
 - UserId 27
 - UserName 27
- GetUserInfo 29**
 - AccountId 29
 - AffiliatedId 30
 - DateTimeCreated 30
 - Email 29
 - EmailVerified 29
 - OMSId 30
 - PasswordHash 29
 - PendingCodeTime 30
 - PendingEmailCode 29
 - RefererId 30
 - Salt 30

- Use2FA 30
- UserId 29
- UserName 29
- GetUserPermissions 31**
 - response an array of
 - permission strings 31
 - UserId 31
- GetUserReportTickets 139**
 - accountIds 141
 - createTime 140
 - initialRunTime 140
 - intervalDuration 141
 - intervalEndTime 140
 - intervalStartTime 140
 - lastInstanceId 141
 - OMSId 140
 - reportFlavor 140
 - ReportFrequency 140
 - RequestId 141
 - RequestingUser 140
 - requestStatus 140
 - UserId 139
- GetUserReportWriterResultRecords 143**
 - descriptorId 143
 - reportDescriptiveHeader 144
 - reportExecutionCompleteTime 144
 - reportExecutionStartTime 144
 - RequestingUser 143
 - resultStatus 143
 - urtTicketId 143
- GetWithdrawalTickets**
 - Limit 229
 - StartIndex 229
- GetWithdrawTemplate 221**
 - AccountId 221
 - BankAccountName 222
 - BankAccountNumber 222
 - BankAddress 222
 - Comment 222 errmsg
 - 222 FullName 222
 - Language 222 OMSId 221
 - ProductId 221
 - result 222
 - statusCode 222
 - SwiftCode 222
 - Template 222
 - templateType 221
- GetWithdrawTemplateTypes 223**
 - AccountId 223
 - errmsg 224
 - OMSId 223
 - ProductId 223
 - result 223
 - statusCode 224
 - TemplateTypes 223
- GetWithdrawTicket 225**
 - AccountId 225, 226
 - Amount 226
 - AssetId 226
 - AssetManagerId 226
 - AssetName 226
 - Attachments 227
 - AuditLog 227
 - BankAccountName 228
 - BankAccountNumber 228
 - BankAddress 228
 - Comment 227
 - Comments 227
 - CreatedTimestamp 227
 - FullName 227
 - Language 227
 - LastUpdateTimeStamp 227
 - OMSId 225, 226
 - OperatorId 225, 226
 - RequestCode 225, 226
 - RequestIP 226
 - RequestUserId 226
 - RequestUserName 226
 - Status 227
 - SwiftCode 228
 - TemplateForm 226
 - TemplateFormType 226
 - TicketNumber 227
 - UpdatedByUser 227
 - UpdatedByUserName 227
- GetWithdrawTickets 229**
 - AccountId 229, 230
 - Amount 230
 - AssetId 230
 - AssetManagerId 230
 - AssetName 230
 - Attachments 232
 - AuditLog 232
 - BankAccountName 232
 - BankAccountNumber 232
 - BankAddress 232
 - Comment 232, 233
 - CommentId 232
 - Comments 232
 - CreatedTimestamp 231
 - EnteredBy 232
 - EnteredDateTime 233
 - FeeAmt 231
 - FullName 232
 - Language 232
 - LastUpdateTimeStamp 231
 - OMSId 229, 231, 233
 - OperatorId 231, 233
 - RequestCode 231
 - RequestIP 231
 - RequestUserId 231
 - RequestUserName 231
 - Status 231
 - SwiftCode 232
 - TemplateForm 230
 - TemplateFormType 231
 - TicketCode 233
 - TicketId 233
- TicketNumber 231
- UpdatedByUser 231
- UpdatedByUserName 231
- GTC (Good 'Til Canceled) 49**
- GUID**
 - CancelUserReport 125
 - globally unique ID string 125
 - Globally Unique ID string 38
 - SetUserInfo 38
- H**
- high 161**
- history**
 - of orders 87, 91, 95
 - ticker 161
- Hold**
 - GetAccountPositions 58
- I**
- IncludeLastCount**
 - SubscribeTicker 177
 - SubscribeTrades 179
- IndsideBidSize**
 - GetOrderHistoryByOrderId 94
 - GetOrdersHistory 99
- initialRunTime**
 - GenerateTradeActivityReport 128
 - GenerateTransactionActivityReport 132
 - GenerateTreasuryActivityReport 136
 - GetUserReportTickets 140
 - ScheduleTradeActivityReport 147
 - ScheduleTransactionActivityReport 151
 - ScheduleTreasuryActivityReport 155
- InProgress**
 - request status 10
- InsideAsk**
 - GetOpenOrders 77
 - GetOpenQuotes 83
 - GetOrderHistory 90
 - GetOrderHistoryByOrderId 94
 - GetOrdersHistory 98
 - GetOrderStatus 104
- InsideAskSize**
 - GetOpenQuotes 83
 - GetOrderHistory 90
 - GetOrderHistoryByOrderId 94
 - GetOrdersHistory 98
 - GetOrderStatus 104

Index

InsideBid

- GetOpenOrders 77
- GetOpenQuotes 83
- GetOrderHistory 90
- GetOrderHistoryByOrderId 94
- GetOrdersHistory 98
- GetOrderStatus 104

InsideBidSize

- GetOpenOrders 77
- GetOpenQuotes 83
- GetOrderHistory 90
- GetOrderStatus 104

instrument

- ticker history 161

Instrument

- equivalent to InstrumentId 76, 82, 88, 92, 97, 102
- GetOpenOrders 76
- GetOpenQuotes 82
- GetOrderHistory 88
- GetOrderHistoryByOrderId 92
- GetOrdersHistory 97
- GetOrderStatus 102

instrument details 67

InstrumentId

- CancelAllOrders 42
- CancelQuote 45
- CancelReplaceOrder 48
- CreateQuote 51
- GetAccountTrades 61
- GetInstrument 67, 68
- GetInstruments 72
- GetL2Snapshot 159
- GetOpenQuotes 79
- GetOrderFee 85
- GetOrdersHistory 95
- GetTickerHistory 161
- GetTradesHistory 164, 165
- ModifyOrder 115
- SendOrder 118
- SubscribeLevel1 171, 172
- SubscribeLevel2 175
- SubscribeTicker 177
- SubscribeTrades 179
- UnsubscribeLevel1 183
- UnsubscribeLevel2 185
- UnsubscribeTicker 187
- UnsubscribeTrades 189
- UpdateQuote 121

instruments

- available on trading venue 71
- defined 4
- Level 2 snapshot 159

InstrumentType

- GetInstrument 68
- GetInstruments 72

Interval

- SubscribeTicker 177

intervalDuration

- GenerateTradeActivityReport 129
- GenerateTransactionActivityReport 133
- GenerateTreasuryActivityReport 137
- GetUserReportTickets 141
- ScheduleTradeActivityReport 145, 147
- ScheduleTransactionActivityReport 149, 151
- ScheduleTreasuryActivityReport 153, 155

intervalEndTime

- GenerateTradeActivityReport 129
- GenerateTransactionActivityReport 133
- GenerateTreasuryActivityReport 137
- GetUserReportTickets 140
- ScheduleTradeActivityReport 147
- ScheduleTransactionActivityReport 151
- ScheduleTreasuryActivityReport 155

intervalStartTime

- GenerateTradeActivityReport 128
- GenerateTransactionActivityReport 132
- GenerateTreasuryActivityReport 136
- GetUserReportTickets 140
- ScheduleTradeActivityReport 147
- ScheduleTransactionActivityReport 151
- ScheduleTreasuryActivityReport 155

Invalid Request

- errorcode 100 3, 198, 214, 237

IOC (Immediate Or Canceled) 49

IsBlockTrade

- GetAccountTrades 61
- GetTradesHistory 166

i sequence number

- message frame 1

IsLockedIn

- GetOpenOrders 77
- GetOpenQuotes 83
- GetOrderHistory 90
- GetOrderHistoryByOrderId 94
- GetOrdersHistory 99
- GetOrderStatus 104

ISO 8601

- time- and date-stamp formats 8

IsQuote

- GetOpenOrders 77
- GetOpenQuotes 83
- GetOrderHistory 90
- GetOrderHistoryByOrderId 94
- GetOrdersHistory 98
- GetOrderStatus 104

J

JavaScript

- and message frame 2

JSON Stringify 2

K

Key

- RemoveUserConfig 33

key/value pairs

- adding custom 35
- custom 27
- verifying 33, 35

L

language

- CreateDepositTicket 194
- GetDepositTicket 219

Language

- GetWithdrawTicket 227
- GetWithdrawTickets 232
- TemplateForm 227, 232

Lanugage

- GetWithdrawTemplate 222

lastInstanceId

- GenerateTradeActivityReport 129
- GenerateTransactionActivityReport 133
- GenerateTreasuryActivityReport 137
- GetUserReportTickets 141
- ScheduleTradeActivityReport 147
- ScheduleTransactionActivityReport 151
- ScheduleTreasuryActivityReport 155

LastName

- GetAllDepositRequestInfoTemplates 201

LastTradedPx

- SubscribeLevel1 172

LastTradedQty

- SubscribeLevel1 172

LastTradePrice

- GetOpenOrders 77
- GetOpenQuotes 83
- GetOrderHistory 90
- GetOrderHistoryByOrderId 94
- GetOrdersHistory 99
- GetOrderStatus 104
- SubscribeLevel2 160, 176

LastTradeTime

SubscribeLevel1 172

LastUpdateTimestamp

GetWithdrawTickets 231

UpdateDepositTicket 236

UpdateWithdrawTicket 241

LastUpdateTimeStamp

GetAllDepositTickets 206

GetDepositTicket 219

GetWithdrawTicket 227

Level 1 and Level 2 Market**Information**

difference between 3

Level 1 information

definition 3

Level 1 Market Data Feed

unsubscribe 183

Level 1 trading information

subscribe to 171

Level 2 information

definition 3

Level 2 Market Data Feed

unsubscribe 185

Level 2 market information

subscribe to 175

Level 2 snapshot 159**Limit**

GetAllDepositTickets 204

GetAllWithdrawTickets 208

GetWithdrawalTickets 229

types of orders 7

LimitOffset

SendOrder 118

LimitPrice

CancelReplaceOrder 48

SendOrder 117

listaccount information for user 111
of account IDs belonging to a user
109

of orders 87

of trades 163

of user report tickets 139

logging out a user 19**LogOut 19**

authenticated calls 3

detail 19

errorcode 19

errmsg 19

result 19

low 161**M****MakerTaker**

GetOrderFee 86

ManualDeposit

template 216

Market

types of orders 7

Market Data Feed

Level 2 unsubscribe 185

trades 179

unsubscribe ticker 187

unsubscribe trades 189

market information

Level 175

MarketStateUpdate

SubscribeAccountEvents 169

trigger event: admin alters market

state 169

matching engine 4**MDUpdateID**

SubscribeLevel2 , 160

MerchantForm

template 216

MerchantRedirect

template 216

message frame

and JavaScript 2

example 2

i sequence number 1

m message type 1

n function name 1

o payload 1

Message Frame

the wrapper of all API calls 1

Microsoft ticks

time- and date-stamp formats 8

m message type

message frame 1

ModifyOrder 115

detail 116

errorcode 116

errmsg 116

InstrumentId 115

OMSID 115

OrderId 115

PreviousOrderRevision 115

Quantity 115

result 116

N**NewOrderRejectEvent**SubscribeAccountEvents 169 trigger
event: order associated with

account is rejected 169

n function name

message frame 1

NoFees

GetProduct 106, 108

Not Authorized

errorcode 203, 198, 214, 237

nsideAskSize

GetOpenOrders 77

O**OMS. See Order Management
System****omsId**

GenerateTradeActivityReport 127

GenerateTransactionActivityReport
131

GenerateTreasuryActivityReport 135

OMSID 121

CancelAllOrders 41

CancelOrder 43

CancelQuote 45

CancelReplaceOrder 48

CreateDepositTicket 193

CreateQuote 51

CreateWithdrawTicket 197

GenerateTradeActivityReport 128

GenerateTransactionActivityReport
132

GenerateTreasuryActivityReport 136

GetAccountInfo 53, 54

GetAccountPositions 57, 58

GetAccountTrades 59, 60

GetAccountTransactions 63, 64

GetAllDepositRequestInfoTemplates
199

GetAllDepositTickets 203, 205

GetAllWithdrawTickets 211

GetAllWithdrawTickets 207, 209

GetDepositInfo 213

GetDepositRequestInfoTemplate 215

GetDepositTicket 217, 218

GetInstrument 67, 68

GetInstruments 71, 72

GetL2Snapshot 159

GetOpenOrders 75, 77

GetOpenQuotes 79, 83

GetOrderFee 85

GetOrderHistory 87, 90

GetOrderHistoryByOrderId 91, 94

- GetOrdersHistory 95, 99
 - GetOrderStatus 101, 104
 - GetProduct 105, 107
 - GetTradesHistory 163, 165
 - GetUserAccountInfos 111
 - GetUserAccounts 109
 - GetUserInfo 30
 - GetUserReportTickets 140
 - GetWithdrawTemplate 221
 - GetWithdrawTemplateTypes 223
 - GetWithdrawTicket 225, 226
 - GetWithdrawTickets 229, 231, 233
 - ModifyOrder 115 sample comment 211, 233
 - ScheduleTradeActivityReport 145, 146
 - ScheduleTransactionActivityReport 149, 150
 - ScheduleTreasuryActivityReport 153, 154
 - SendOrder 118
 - SetUserInfo 38
 - SubscribeAccountEvents 167
 - SubscribeLevel1 171, 172
 - SubscribeLevel2 175
 - SubscribeTicker 177
 - SubscribeTrades 179, 180
 - UnsubscribeLevel1 183
 - UnsubscribeLevel2 185
 - UnsubscribeTicker 187
 - UnsubscribeTrades 189
 - UpdateDepositTicket 236
 - UpdateQuote 121
 - UpdateWithdrawTicket 240
- OMSID**
- GetUserAccountInfos 112
- one order cancels another 117**
- o payload**
- message frame 1
- open 161**
- open orders**
- cancel 43
 - list of 75
- Operation Failed**
- errorcode 101 3, 198, 214, 237
- OperatorId**
- CreateDepositTicket 193
 - GetAllDepositTickets 203, 205
 - GetAllWithdrawTickets 211
 - GetAllWithdrawTickets 207, 209
 - GetDepositTicket 217, 218
 - GetWithdrawTicket 225, 226
 - GetWithdrawTickets 231, 233
 - RegisterNewUser 13
 - sample comment 211, 233
 - UpdateDepositTicket 236
 - UpdateWithdrawTicket 240
- order**
- cancel 43
 - cancel all 41
 - cancel and replace 47 change an existing order 115 create and submit 117 get status 101
 - history 87
- Order1**
- SubscribeTrades 180
- Order1ClientId**
- SubscribeTrades 181
- Order1Side**
- SubscribeTrades 181
- Order2**
- SubscribeTrades 180
- Order2ClientId**
- SubscribeTrades 181
- Order2Side**
- SubscribeTrades 181
- order book 47**
- modifying an order 115
 - update quote 121
- OrderFee**
- GetOrderFee 86
- OrderId**
- CancelOrder 43
 - GetAccountTrades 60
 - GetOpenOrders 76
 - GetOpenQuotes 82
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 91, 92
 - GetOrdersHistory 97
 - GetOrderStatus 101, 102
 - GetTradesHistory 164, 165
 - ModifyOrder 115
 - SendOrder 119
- OrderIdOCO**
- CancelReplaceOrder 49
 - SendOrder 117
- OrderIdToReplace**
- CancelReplaceOrder 48
- Order Management System**
- definition 4
- OrderOriginator**
- GetAccountTrades 60
 - GetTradesHistory 165
- orders**
- history 91, 95
 - list of open orders 75
 - multiple history 95
 - subscribing to 167
- Orders**
- defined 5
 - SubscribeLevel2 160, 176
- OrderState**
- GetOpenOrders 77
 - GetOpenQuotes 82
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
 - GetOrderStatus 103
- OrderStateEvent**
- SubscribeAccountEvents 169 trigger event: status changes for an order 169
- OrderTradeEvent**
- SubscribeAccountEvents 170
 - trigger event: an order from your account results in trade 170
- OrderTradeRevision**
- GetAccountTrades 61
 - GetTradesHistory 166
- OrderType**
- CancelReplaceOrder 48
 - GetOpenOrders 76
 - GetOpenQuotes 82
 - GetOrderFee 86
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 97
 - GetOrderStatus 103 SendOrder 118
- Order types 7**
- OrigClOrdId**
- CancelReplaceOrder 49
 - GetOpenQuotes 83
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
- OrigClOrdId**
- GetOrderStatus 103
- OriginalOrderId**
- GetOrdersHistory 95
- OrigOrderId**
- CancelReplaceOrder 49
 - GetOpenOrders 77
 - GetOpenQuotes 83
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
- OrigOrderID**
- GetOrderStatus 103
- OrigQuantity**
- GetOpenOrders 77
 - GetOpenQuotes 82
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
 - GetOrderStatus 103

P

ParentID

GetAccountInfo 54
GetUserAccountInfos 112

password

reset 21

Password

SetUserInfo 37
WebAuthenticateUser 15

PasswordHash

GetUserInfo 29
RegisterNewUser 13
SetUserInfo 38

PegPriceType

CancelReplaceOrder 48
SendOrder 118

PendingCodeTime

GetUserInfo 30
SetUserInfo 38

PendingDeposits

GetAccountPositions 58

PendingDepositUpdate

SubscribeAccountEvents 170
trigger event: a deposit pending on
your account 170

PendingEmailCode

GetUserInfo 29
SetUserInfo 38

PendingWithdraws

GetAccountPositions 58

permissions

how they're set 4

Permissions

list of available permissions 25

positions 57

position updates

subscribing to 167

POSIX

time- and date-stamp formats 8

PreviousOrderRevision

ModifyOrder 115

PreviousSessionStatus

GetInstrument 68
GetInstruments 73

Price

GetAccountTrades 61
GetOpenOrders 76
GetOpenQuotes 82
GetOrderFee 85
GetOrderHistory 88
GetOrderHistoryByOrderId 92
GetOrdersHistory 97

GetOrderStatus 102
GetTradesHistory 166
SubscribeLevel2 160, 176
SubscribeTrades 180

priority

in order book 115

ProcessInfo

GetAllDepositRequestInfoTemplates
200
GetDepositRequestInfoTemplate 216

product

available on trading venue 107
details 105

Product

GetProduct 106, 108
same as Asset 4

Product1

GetInstrument 68
GetInstruments 72

Product1Symbol

GetInstrument 68
GetInstruments 72

Product2

GetInstrument 68
GetInstruments 72

Product2Symbol

GetInstrument 68
GetInstruments 72

ProductFullName

GetProduct 106, 108

ProductId

CreateWithdrawTicket 197
GetAccountPositions 58
GetAccountTransactions 64
GetAllDepositRequestInfoTemplates
199
GetDepositInfo 213
GetDepositRequestInfoTemplate 215
GetOrderFee 85, 86
GetProduct 105, 107
GetWithdrawTemplate 221
GetWithdrawTemplateTypes 223

ProductPairCode

SubscribeLevel2 160, 176
SubscribeTrades 180

Products and Instruments

defined 4

ProductSymbol

GetAccountPositions 58

ProductType

GetProduct 106, 108

providerId

GetAllDepositRequestInfoTemplates
200

ProviderId

GetDepositInfo 214

providerName

GetAllDepositRequestInfoTemplates
200

ProviderType

GetDepositRequestInfoTemplate 216

public calls and authenticated

calls 3

Q

Quantity

CancelReplaceOrder 49
GetAccountTrades 61
GetOpenOrders 76
GetOpenQuotes 82
GetOrderHistory 88
GetOrderHistoryByOrderId 92
GetOrdersHistory 97
GetOrderStatus 102
GetTradesHistory 165
ModifyOrder 115
SendOrder 117
SubscribeLevel2 160, 176
SubscribeTrades 180

QuantityExecuted

GetOpenOrders 77
GetOpenQuotes 82
GetOrderHistory 89
GetOrderHistoryByOrderId 93
GetOrdersHistory 98
GetOrderStatus 103

QuantityIncrement

GetInstrument 69
GetInstruments 73

quote

and registered market participants
45, 121
canceling 45
creating 51
obtaining quotes 79
update 121

Quotes and Orders

defined 5

R

ReceiveTime

GetOpenOrders 77
GetOpenQuotes 82
GetOrderHistory 89
GetOrderHistoryByOrderId 93
GetOrdersHistory 98

Index

- GetOrderStatus 103
- ReceiveTimeTicks**
 - GetOpenOrders 77
 - GetOpenQuotes 82
 - GetOrderHistory 89
 - GetOrderHistoryByOrderId 93
 - GetOrdersHistory 98
 - GetOrderStatus 103
- ReferenceId**
 - GetAccountTransactions 64
- ReferenceType**
 - GetAccountTransactions 64
- RefererId**
 - GetAccountInfo 55
 - GetUserAccountInfos 113
 - GetUserInfo 30
 - SetUserInfo 38
- RegisterNewUser 13**
 - AffiliatedId 13
 - Email 13
 - OperatorId 13
 - PasswordHash 13
 - UserConfig 13
 - UserId 14
 - userInfo 13
 - UserName 13
- RejectReason**
 - GetOpenOrders 77
 - GetOpenQuotes 83
 - GetOrderHistory 90
 - GetOrderHistoryByOrderId 94
 - GetOrdersHistory 99
 - GetOrderStatus 104
- RemainingQuantity**
 - GetAccountTrades 61
 - GetTradesHistory 165
- RemoveUserConfig 33**
 - detail 34
 - errorcode 34
 - errmsg 34
 - Key 33
 - result 34
 - UserId 33
 - UserName 33
- ReplacementCIOrdId**
 - CancelReplaceOrder 49
- ReplacementOrderId**
 - CancelReplaceOrder 49
- replacing an order 47**
- report**
 - scheduling - trade activity 145
- reportDescriptiveHeader**
 - GetUserReportWriterResultRecords 144
- reportExecutionCompleteTime**
 - GetUserReportWriterResultRecords 144
- reportExecutionStartTime**
 - GetUserReportWriterResultRecords 144
- reportFlavor**
 - GenerateTradeActivityReport 128
 - GenerateTransactionActivityReport 132
 - GenerateTreasuryActivityReport 136
 - GetUserReportTickets 140
 - ScheduleTradeActivityReport 146
 - ScheduleTransactionActivityReport 150
 - ScheduleTreasuryActivityReport 154
- ReportFrequency**
 - GenerateTradeActivityReport 129
 - GenerateTransactionActivityReport 133
 - GenerateTreasuryActivityReport 137
 - GetUserReportTickets 140
 - ScheduleTradeActivityReport 147
 - ScheduleTransactionActivityReport 151
 - ScheduleTreasuryActivityReport 155
- reports**
 - identifying 139
 - list of tickets 139
 - on demand 127
 - schedule a transaction activity report 149
 - trade activity 127, 145
 - transaction activity 131
 - treasury activity 135, 153
- report status 143**
- report type**
 - defined 9
 - Trade Activity 9
 - Transaction 9
 - Treasury 9
- requestcode**
 - CreateDepositTicket 194
- RequestCode**
 - GetAllDepositTickets 205
 - GetAllWithdrawTickets 209
 - GetDepositTicket 217, 218
 - GetWithdrawTicket 225, 226
 - GetWithdrawTickets 231
 - UpdateDepositTicket 236
 - UpdateWithdrawTicket 240
- RequestId**
 - GenerateTradeActivityReport 129
 - GenerateTransactionActivityReport 133
 - GenerateTreasuryActivityReport 137
 - GetUserReportTickets 141
 - ScheduleTradeActivityReport 147
- ScheduleTransactionActivityReport** 151
- ScheduleTreasuryActivityReport** 155
- RequestingUser**
 - GenerateTradeActivityReport 128
 - GenerateTransactionActivityReport 132
 - GenerateTreasuryActivityReport 136
 - GetUserReportTickets 140
 - GetUserReportWriterResultRecords 143
 - ScheduleTradeActivityReport 146
 - ScheduleTransactionActivityReport 150
 - ScheduleTreasuryActivityReport 154
- RequestIp**
 - GetAllDepositTickets 205
- RequestIP**
 - GetAllWithdrawTickets 209
 - GetDepositTicket 218
 - GetWithdrawTicket 226
 - GetWithdrawTickets 231
 - UpdateDepositTicket 236
 - UpdateWithdrawTicket 240
- request status 10**
 - Aborted 10
 - Aborting 10
 - Completed 10
 - InProgress 10
 - Scheduled 10
 - Submitted 10
 - SysRetired 10
 - UserCanceled 10
 - UserCanceledPending 10
 - Validating 10
- requestStatus**
 - GenerateTradeActivityReport 129
 - GenerateTransactionActivityReport 133
 - GenerateTreasuryActivityReport 137
 - GetUserReportTickets 140
 - ScheduleTradeActivityReport 147
 - ScheduleTransactionActivityReport 151
 - ScheduleTreasuryActivityReport 155
- RequestUser**
 - CreateDepositTicket 193
 - GetAllDepositTickets 205
 - GetDepositTicket 218
 - UpdateDepositTicket 236
- RequestUserId**
 - GetAllWithdrawTickets 209
 - GetWithdrawTicket 226
 - GetWithdrawTickets 231
 - UpdateWithdrawTicket 240
- RequestUserName**
 - GetAllDepositTickets 205
 - GetAllWithdrawTickets 209
 - GetDepositTicket 218

- GetWithdrawTicket 226
- GetWithdrawTickets 231
- UpdateDepositTicket 236
- UpdateWithdrawTicket 240
- reset password 21**
- ResetPassword 21**
 - result 21
 - UserName 21
- Resource Not Found**
 - errorCode 104 3, 198, 214, 237
- response object, generic 2**
- result**
 - CancelAllOrders 42
 - CancelOrder 44
 - CancelQuote 46
 - CancelUserReport 125
 - CreateQuote 52
 - CreateWithdrawTicket 198
 - GetAllDepositRequestInfoTemplates 200
 - GetDepositInfo 214
 - GetDepositRequestInfoTemplate 216
 - GetWithdrawTemplate 222
 - GetWithdrawTemplateTypes 223
 - in generic response object 2, 198
 - LogOut 19
 - ModifyOrder 116
 - RemoveUserConfig 34
 - ResetPassword 21
 - SetUserConfig 36
 - UnsubscribeLevel1 183
 - UnsubscribeLevel2 185
 - UnsubscribeTicker 187
 - UnsubscribeTrades 189
 - UpdateDepositTicket 237
 - UpdateQuote 122
 - UpdateWithdrawTicket 241
- resultStatus**
 - GetUserReportWriterResultRecords 143
- return list of withdraw tickets 207, 229**
- RiskType**
 - GetAccountInfo 54
 - GetUserAccountInfos 112
- Rolling24HrNumTrades**
 - SubscribeLevel1 172
- Rolling24HrPxChange**
 - SubscribeLevel1 172
- Rolling24HrVolume**
 - SubscribeLevel1 172

S

- Salt**
 - GetUserInfo 30
 - SetUserInfo 38
- sample comment 211, 232**
- Scheduled**
 - request status 10
- ScheduleTradeActivityReport 145**
 - AccountIdList 145
 - accountIds 147
 - beginTime 145
 - createTime 147 frequency 145 initialRunTime 147
 - intervalDuration 145, 147
 - intervalEndTime 147
 - intervalStartTime 147
 - lastInstanceId 147 OMSId 145, 146 reportFlavor 146
 - ReportFrequency 147
 - RequestId 147
 - RequestingUser 146
 - requestStatus 147
- ScheduleTransactionActivityReport 149**
 - AccountIdList 149
 - accountIds 151
 - beginTime 149
 - createTime 151 frequency 149 initialRunTime 151
 - intervalDuration 149, 151
 - intervalEndTime 151
 - intervalStartTime 151
 - lastInstanceId 151 OMSId 149, 150 reportFlavor 150
 - ReportFrequency 151
 - RequestId 151
 - RequestingUser 150
 - requestStatus 151
- ScheduleTreasuryActivityReport 153**
 - AccountIdList 153
 - accountIds 155
 - beginTime 153
 - createTime 155 frequency 153 initialRunTime 155
 - intervalDuration 153, 155
 - intervalEndTime 155
 - intervalStartTime 155
 - lastInstanceId 155 OMSId 153, 154 reportFlavor 154
 - ReportFrequency 155
 - RequestId 155

- RequestingUser 154
- requestStatus 155
- scheduling a trade activity report 145**
- scheduling a transaction activity report 149**
- SelfTradePrevention**
 - GetInstrument 69
 - GetInstruments 73
- SendOrder 117**
 - AccountId 117
 - ClientOrderId 117
 - DisplayQuantity 117
 - errorMsg 119
 - InstrumentId 118
 - LimitOffset 118
 - LimitPrice 117
 - OMSId 118 OrderId 119 OrderIdOCO 117 OrderType 118
 - PegPriceType 118
 - Quantity 117 Side 118
 - status 119
 - StopPrice 118
 - TimeInForce 118
 - TrailingAmount 118
 - UseDisplayQuantity 117
- Server Error**
 - errorCode 102 3, 198, 214, 237
- SessionClose**
 - SubscribeLevel1 172
- SessionHigh**
 - SubscribeLevel1 172
- SessionLow**
 - SubscribeLevel1 172
- SessionOpen**
 - SubscribeLevel1 172
- SessionStatus**
 - GetInstrument 68
 - GetInstruments 72
- SessionStatusDateTime**
 - GetInstrument 69
 - GetInstruments 73
- SessionToken**
 - Authenticate2FA 17
 - WebAuthenticateUser 16
- SetUserConfig 35**
 - Config 35
 - detail 36
 - errorCode 36
 - errorMsg 36
 - result 36
 - UserId 35
 - UserName 35

Index

SetUserInfo 37

- AccountId 37, 38
- AffiliatedId 38
- DateTimeCreated 38
- Email 37, 38
- EmailVerified 37, 38
- enter user information 37
- GUID 38
- OMSIId 38
- Password 37
- PasswordHash 38
- PendingCodeTime 38
- PendingEmailCode 38
- RefererId 38
- Salt 38
- Use2FA 37, 38
- UserId 37, 38
- UserName 37, 38

Side

- CancelReplaceOrder 48
- GetAccountTrades 61
- GetOpenOrders 76
- GetOpenQuotes 81
- GetOrderHistory 88
- GetOrderHistoryByOrderId 92
- GetOrdersHistory 97
- GetOrderStatus 102
- GetTradesHistory 165
- SendOrder 118
- SubscribeLevel2 , 160

Side1AccountId

- SubscribeTrades 180

Side2AccountId

- SubscribeTrades 180

SortIndex

- GetInstrument 68
- GetInstruments 72

StartIndex

- GetAccountTrades 59
- GetAllDepositTickets 204
- GetAllWithdrawTickets 208
- GetOrdersHistory 96
- GetTradesHistory 164
- GetWithdrawalTickets 229

startTime

- GenerateTradeActivityReport 127
- GenerateTransactionActivityReport 131
- GenerateTreasuryActivityReport 135

StartTimestamp

- GetAllDepositTickets 204
- GetAllWithdrawTickets 208
- GetOrdersHistory 96

StartTimeStamp

- GetTradesHistory 164

status

- SendOrder 119

Status

- CreateDepositTicket 194
- GetAllDepositTickets 203, 205
- GetAllWithdrawTickets 207, 210
- GetDepositTicket 218
- GetWithdrawTicket 227
- GetWithdrawTickets 231
- UpdateDepositTicket 236
- UpdateWithdrawTicket 240

statuscode

- GetAllDepositRequestInfoTemplates 200
- GetDepositInfo 214
- GetDepositRequestInfoTemplate 216
- GetWithdrawTemplate 222
- GetWithdrawTemplateTypes 224 see errorcode 214

status of an order 101

StopLimit

- types of orders 7

StopMarket

- types of orders 7

StopPrice

- CancelReplaceOrder 48
- SendOrder 118

SubAccountId

- GetAccountTrades 60
- GetTradesHistory 165

submit an order 117

Submitted

- request status 10

subscribable feeds

- access controlled by venue operator 3

Subscribe

- SubscribeAccountEvents 167

SubscribeAccountEvents 167

- AccountId 167
- AccountPositionEvent 168
- CancelAllOrdersRejectEvent 168
- CancelOrderRejectEvent 168
- CancelReplaceOrderRejectEvent 169
- MarketStateUpdate 169
- NewOrderRejectEvent 169
- OMSIId 167
- OrderStateEvent 169
- OrderTradeEvent 170
- PendingDepositUpdate 170
- Subscribe 167

SubscribeLevel1 171

- authenticated calls 3
- BestBid 172
- BestOffer 172
- CurrentDayNumTrades 172
- CurrentDayPxChange 172
- CurrentDayVolume 172
- InstrumentId 171, 172

- LastTradedPx 172
- LastTradedQty 172
- LastTradeTime 172
- OMSIId 171, 172
- Rolling24HrNumTrades 172
- Rolling24HrPxChange 172
- Rolling24HrVolume 172
- SessionClose 172
- SessionHigh 172
- SessionLow 172
- SessionOpen 172
- Symbol 171
- TimeStamp 173
- Volume 172

SubscribeLevel2 175

- Accounts 160, 176
- ActionDateTime , 160
- ActionType 160, 176
- authenticated calls 3
- Depth. *used as depth of market* AccountId
- InstrumentId 175
- LastTradePrice 160, 176
- MDUpdateID , 160
- OMSIId 175
- Orders 160, 176
- Price 160, 176
- ProductPairCode 160, 176
- Quantity 160, 176
- Side , 160
- Symbol 175

SubscribeTicker 177

- authenticated calls 3
- IncludeLastCount 177
- InstrumentId 177
- Interval 177
- OMSIId 177
- response format shown with strings 177

SubscribeTrades 179

- BlockTrade 181
- Direction 180
- IncludeLastCount 179
- InstrumentId 179
- OMSIId 179, 180
- Order1 180
- Order1ClientId 181
- Order1Side 181
- Order2 180
- Order2ClientId 181
- Order2Side 181
- Price 180
- ProductPairCode 180
- Quantity 180
- Side1AccountId 180
- Side2AccountId 180
- TakerSide 180
- TradeID 180
- TradeTime 180

subscribing

- to deposits 167

- to orders 167
- to position updates 167
- to trades 167
- to withdrawals 167

subscription

- Level 2 market information 175

subscription to data feed

- unsubscribe 183

success

- CreateDepositTicket 194

SupportedVenueIds

- GetAccountInfo 55

SwiftCode

- GetWithdrawTemplate 222
- GetWithdrawTicket 228, 232
- TemplateForm 228, 232

Symbol

- GetInstrument 68
- GetInstruments 72
- SubscribeLevel1 171
- SubscribeLevel2 175

SysRetired

- request status 10

T**TakerSide**

- SubscribeTrades 180

template

- example 211
- find withdrawal template 223
- return template text 221
- withdrawal 223

Template

- GetDepositRequestInfoTemplate 216
- GetWithdrawTemplate 222

template, deposit

- GetDepositRequestInfoTemplate 215

Templateform

- Comment 227, 232

TemplateForm

- BankAccountName 228, 232
- BankAccountNumber 228, 232
- BankAddress 228, 232
- CreateWithdrawTicket 197
- FullName 227, 232
- GetAllWithdrawTickets 209
- GetWithdrawTicket 226
- GetWithdrawTickets 230
- Language 227, 232
- SwiftCode 228, 232
- UpdateWithdrawTicket 240

TemplateFormType

- GetAllWithdrawTickets 209

- GetWithdrawTicket 226
- GetWithdrawTickets 231
- UpdateWithdrawTicket 240

Templates

- GetAllDepositRequestInfoTemplates 200

templates, deposit

- GetAllDepositRequestInfoTemplates 199

templateType

- GetWithdrawTemplate 221

TemplateType

- CreateWithdrawTicket 197, 198
- example template 211
- GetAllWithdrawTickets 211
- UpdateWithdrawTicket 241

TemplateTypes

- GetWithdrawTemplateTypes 223

template, withdrawal

- specifying 197

ticker

- subscribe 177
- unsubscribe 187

ticker history 161**ticker information**

- Level 2 175

Ticker Market Data Feed 177

- unsubscribe 187

TicketCode

- GetAllWithdrawTickets 211
- GetWithdrawTickets 233
- sample comment 211, 233

TicketId

- GetAllDepositTickets 203
- GetAllWithdrawTickets 211
- GetAllWithdrawTickets 207
- GetWithdrawTickets 233
- sample comment 211, 233

TicketNumber

- GetAllDepositTickets 205
- GetAllWithdrawTickets 210
- GetDepositTicket 219
- GetWithdrawTicket 227
- GetWithdrawTickets 231
- UpdateDepositTicket 236
- UpdateWithdrawTicket 240

TickSize

- GetProduct 106, 108

time- and date-stamp formats 8

- ISO 8601 8
- Microsoft ticks 8
- POSIX 8

TimeInForce

- CancelReplaceOrder 49
- SendOrder 118

TimeStamp

- GetAccountTransactions 64
- SubscribeLevel1 173

TotalDayDeposits

- GetAccountPositions 58

TotalDayWithdraws

- GetAccountPositions 58

TotalMonthWithdraws

- GetAccountPositions 58

trade

- subscribe to Level 1 171

Trade Activity

- report type 9

trade activity report 127, 145**TradeId**

- GetAccountTrades 60
- GetTradesHistory 164, 165

TradeID

- SubscribeTrades 180

trades 127

- by account 59 market data
- feed 179 obtaining list of
- 163 subscribe to 167, 179
- unsubscribe data feed 189

Trades Market Data Feed

- unsubscribe 189

TradeTime

- GetAccountTrades 61
- GetTradesHistory 166
- SubscribeTrades 180

TradeTimeMS

- GetAccountTrades 60
- GetTradesHistory 165

trading day

- and 24-hour exchanges 9
- UTC midnight to UTC midnight 9

trading venue

- and access to ticker and
- subscribable feeds 3
- definition 4

TrailingAmount

- CancelReplaceOrder 48
- SendOrder 118

TrailingStopMarket

- types of orders 7

Transaction

- report type 9

transaction activity report 131

- scheduling 149

TransactionId

- GetAccountTransactions 64

Index

transactions

list of 63

TransactionType

GetAccountTransactions 64

Treasury

report type 9

treasury activity report 135

scheduling 153

trigger event: account

balance changes

AccountPositionEvent 168

trigger event: a deposit

pending on your account

PendingDepositUpdate 170

trigger event: admin alters market state

MarketStateUpdate 169

trigger event: all orders rejected

CancelAllOrdersRejectEvent 168

trigger event: an order from your account results in trade

OrderTradeEvent 170

trigger event: order associated with account is rejected

NewOrderRejectEvent 169

trigger event: order is canceled

CancelOrderRejectEvent 168

trigger event: order rejected

even with cancel-replace

CancelReplaceOrderRejectEvent 169

trigger event: status changes for an order

OrderStateEvent 169

two-factor authentication 17

types of orders

BlockTrade 7

Limit 7

Market 7

StopLimit 7

StopMarket 7

TrailingStopLimit 7

TrailingStopMarket 7

Unknown 7

U

Unknown

types of orders 7

UnsubscribeLevel1 183

authenticated calls 3

detail 183

errorcode 183

errormsg 183

InstrumentId 183

OMSid 183

result 183

UnsubscribeLevel2 185

authenticated calls 3

detail 185

errorcode 185

errormsg 185

InstrumentId 185

OMSid 185

result 185

UnsubscribeTicker 187

authenticated calls 3

detail 187

errorcode 187

errormsg 187

InstrumentId 187

OMSid 187

result 187

UnsubscribeTrades 189

detail 189

errorcode 189

errormsg 189

InstrumentId 189

OMSid 189

result 189

update a quote 121

UpdateByUserName

GetAllDepositTickets 205

UpdatedByUser

GetAllDepositTickets 205

GetAllWithdrawTickets 210

GetDepositTicket 218

GetWithdrawTicket 227

GetWithdrawTickets 231

UpdateDepositTicket 236

UpdateWithdrawTicket 240

UpdatedByUserName

GetAllWithdrawTickets 210

GetDepositTicket 219

GetWithdrawTicket 227

GetWithdrawTickets 231

UpdateDepositTicket 236

UpdateWithdrawTicket 240

UpdateDepositTicket 235

AccountId 235

Amount 235

AssetId 235

AssetManagerId 235

AssetName 235

Attachments 236

Comments 236

CreatedTimestamp 236

DepositInfo 236

detail 237

errorcode 237

errormsg 237

FeeAmt 236

LastUpdateTimestamp 236

OMSid 236

OperatorId 236

RequestCode 236

RequestIP 236

RequestUser 236

RequestUserName 236

result 237

Status 236

TicketNumber 236

UpdatedByUser 236

UpdatedByUserName 236

UpdateQuote 121

AdkQuoteId 122

AskResult 122

BidQuoteId 122

BidResult 122

detail 122

errorcode 122

result 122

UpdateWithdrawTicket 239

AccountId 239

Amount 240

AssetId 239

AssetManagerId 239

AssetName 240

Attachments 241

AuditLog 241

Comment 241

Comments 241

CreatedTimestamp 241

detail 242

errorcode 242

errormsg 242

ExternalAddress 241

FeeAmt 240

LastUpdateTimestamp 241

OMSid 240

OperatorId 240

RequestCode 240

RequestIP 240

RequestUserId 240

RequestUserName 240

result 241

Status 240

TemplateForm 240

TemplateFormType 240

TemplateType 241

TicketNumber 240

UpdatedByUser 240

UpdatedByUserName 240

urtTicketId

GetUserReportWriterResultRecords
143

Use2FA

GetUserInfo 30

SetUserInfo 37, 38

UseDisplayQuantity

SendOrder 117

UseGetDepositWorkflow

GetAllDepositRequestInfoTemplates
200

GetDepositRequestInfoTemplate 216

user

authenticating 15
create new 13
logging out 19

UserCanceled

request status 10

UserCanceledPending

request status 10

UserConfig

RegisterNewUser 13

UserId

CancelAllOrders 41
GetOrdersHistory 95
GetTradesHistory 109, 164
GetUserAccountInfos 111
GetUserAccounts 109
GetUserConfig 27
GetUserInfo 29
GetUserPermissions 31
GetUserReportTickets 139
RemoveUserConfig 33
SetUserConfig 35
SetUserInfo 37, 38
WebAuthenticateUser 16

UserID

RegisterNewUser 14

userInfo

RegisterNewUser 13

user information

adding 35
deleting 33
SetUserInfo 37
storing 27

UserName

GetAllDepositTickets 204
GetAllWithdrawTickets 208
GetUserAccountInfos 111
GetUserAccounts 109
GetUserConfig 27
GetUserInfo 29
RegisterNewUser 13
RemoveUserConfig 33
ResetPassword 21
SetUserConfig 35
SetUserInfo 37, 38
WebAuthenticateUser 15

user password

reset 21

user permissions

common sets of permissions 31
getting a list 31

UserReport

CancelUserReport 125

user reports

canceling 125

user report tickets

get list of 139

users

may have multiple accounts 4

users and multiple accounts 4**V****Validating**

request status 10

Value

GetAccountTrades 61
GetTradesHistory 166

VenueId

GetInstrument 68
GetInstruments 72

VenueInstrumentId

GetInstrument 68
GetInstruments 72

VerificationLevel

GetAccountInfo 54
GetUserAccountInfos 112

volume 161**Volume**

SubscribeLevel1 172

W**WebAuthenticateUser 15**

Authenticated 15
authenticated calls 3
Password 15
SessionToken 16
UserId 16
UserName 15

withdrawal

updating 239

withdrawals

obtaining list of 207, 229
require admin approval 197

withdrawals, tracked by system

193

withdrawal template

return template text 221
specifying 197

withdrawal templates

find appropriate templates 223

withdrawals

subscribing to 167

withdraw ticket

find a single withdraw ticket 225
updating 239

wrap calls in a message frame 1