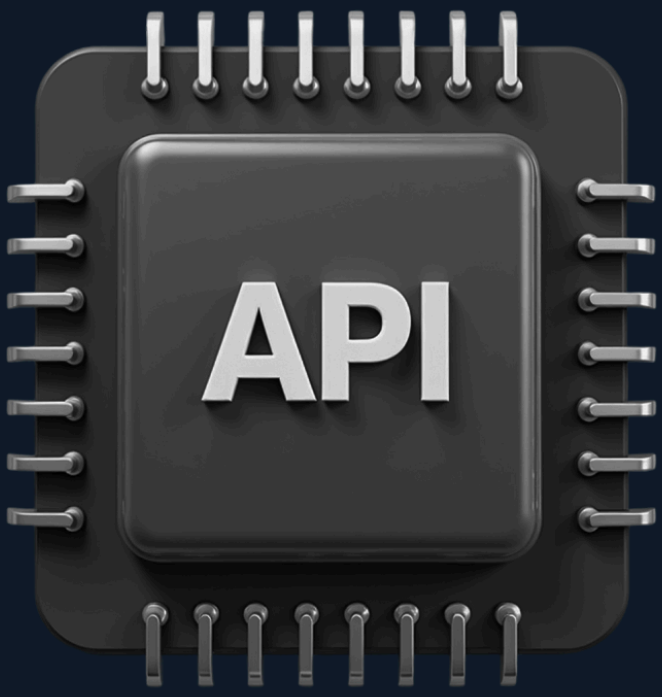
А Инвестиции



Альфа-Инвестиции

PRO WebSocket API

1. Подключение

Протокол: WebSocket

EndPoint: `ws://127.0.0.1:3366/router/`

Условие: терминал должен быть запущен.

2. Алгоритм работы

Подключение устанавливается с роутером, который обеспечивает маршрутизацию сообщений между клиентами. Сервисы, предоставляющие те или иные функции, точно также подключаются к роутеру.

На уровне роутера есть два вида сообщений — `RoutingRequest` и `RoutingError`. Все сообщения, отправляемые роутеру — это `RoutingRequest`-ы. От роутера могут приходить `RoutingRequest`-ы, отправленные другими клиентами, либо `RoutingError` в случае ошибок, когда роутер не смог обработать и/или смаршрутизировать `RoutingRequest`.

В алгоритме маршрутизации, роутер оперирует каналами (Channel). Для того чтобы получать сообщения, отправленные в какой-то канал, нужно подписаться на этот канал. Есть два вида подписки – шина (Bus) и сервис (Responder). На приём сообщений с канала шины могут подписаться несколько клиентов роутера. На приём сообщений с канала сервиса может подписаться только один, он и будет обязан отвечать на запросы, приходящие ему, отправляя ответы в тот же канал — роутер доставит ответ тому клиенту, который отправил запрос к сервису.

Для отправки сообщений в канал быть его подписчиком не обязательно.

2.1 `RoutingRequest`

Поля, описанные ниже, отправляются в запросе и так же приходят в ответе на этот запрос.

```
{
  "Command": "string: Команда роутеру",
  "Channel": "string: Канал",
  "Payload": "json?: Контент",
  "Id": "string?: Идентификатор запроса"
}
```

Обязательны поля `Command` и `Channel`. Остальные заполняются или не заполняются в зависимости от команды. По полю `Id` проверяется соответствие запроса и ответа.

Команды `RoutingRequest.Command`

`listen` **— Подписка на шину.**
С этого момента, все сообщения, отправляемые на указанный в запросе канал, будут доставляться клиенту.
Ошибок не предполагается, в качестве канала может использоваться любой текст.

`unlisten` **— Отписка от шины.**
С этого момента, сообщения, отправляемые на указанный в запросе канал, перестанут доставляться клиенту.
Ошибок не предполагается, даже если подписки не было.

`broadcast` **— Отправка контента на шину.**
Контент отправляется подписчикам канала. Если таковых нет, ничего не происходит, ошибок также не предполагается.

`register` **— Регистрация сервиса (респондера).**
Ошибка может быть, если на указанном сервисе уже зарегистрирован другой респондер.
С момента регистрации, запросы клиентов в канал будут отправляться зарегистрировавшемуся сервису.

`unregister` **— Отмена регистрации сервиса (респондера).**
После отмены регистрации, клиент перестаёт выполнять роль сервиса на указанном канале.

`request` **— Запрос сервису.**
Запрос клиента, который маршрутизируется сервису. Должен быть указан `Id`, чтобы сопоставлять запрос клиента с ответом.
Ошибка может возникнуть, если на указанном канале респондер не зарегистрирован.

`response` **— Ответ сервиса на запрос.**
Ответ сервиса, который маршрутизируется клиенту. Должен быть указан `Id` из запроса, чтобы роутер мог сопоставить его с клиентом, приславшим запрос.
Ошибка может возникнуть, если по указанному идентификатору не найден клиент и его запрос.

2.2 `RoutingError`

Приходит в ответе на отправленный запрос.

```
{
  "Code": "int: Код ошибки",
  "Message": "string?: Текст сообщения"
}
```

Возможные коды ошибок:

- 0 — нет ошибки,
- 5 — не найдено.

3. Сервис публикации данных

Табличные данные хранятся во внутреннем хранилище терминала в виде записей с ключом типа int64. Для доступа к ним, в терминале работает сервис публикации, который публикует изменения данных на каналах типа шина.

Для получения табличных данных определённого типа, необходимо отправить listen-запрос на роутер, на соответствующий канал, имя которого определяется по типу записи: `#Data.Bus.<EntityType>`.
Например, записи типа ObjectEntity, публикуется на канале `#Data.Bus.ObjectEntity`.
Тогда запрос к роутеру будет выглядеть так:

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.ObjectEntity"
}
```

Соответственно, для прекращения приёма обновлений данных, нужно отписаться от соответствующего канала командой роутера `unlisten`.

```
{
  "Command": "unlisten",
  "Channel": "#Data.Bus.ObjectEntity"
}
```

На шину данные отправляются в виде следующего запроса к роутеру:

```
{
  "Command": "broadcast",
  "Channel": "#Data.Bus.ObjectEntity",
  "Payload":
```

```
  {
    "Type": "ObjectEntity",
    "Deleted": [{}],
    "Updated": [{}]
  }
}
```

Т.е. публикуется тип записи, и сами записи, отдельно удалённые, в массиве `"Deleted"`, и отдельно обновлённые или добавленные, в массиве `"Updated"`.

3.1 `DataEntity`

Приходит в ответе на запрос данных. Общие для всех записей с данными поля:

```
{
  "DataId": "int64: Первичный ключ в таблице хранилища данных",
  "Operation": "int: Вид операции",
  "Version": "int64: Версия данных"
}
```

Пример одной записи для MarketBoardEntity (из Payload).

```
{
   "data":{
      "DataCompressed":null,
      "Expires":"2024-07-30T07:44:29.2319297Z",
      "Id":24,
      "Subscribed":true,
      "Type":"MarketBoardEntity",
      "Data":[
         {
            "IdMarketBoard":11,
            "IdMarketType":2,
            "IdGate":2,
            "IdBoardType":1,
            "CodeMarketBoard":"MICI",
            "NameMarketBoard":"ММВБ Индексы",
            ...
            "UniversalMarketCode":"MOEX_INDICES",
            "DataId":11,
```

```
            "Version":140265814897,
            "Operation":2
         }
      ]
   }
}
```

Вид операции принимает одно из значений:

- 0 — новая запись
- 1 — удалённая запись
- 2 — обновлённая запись

Версия данных в будущем позволит подписываться на данные с указанием версии уже имеющихся у клиента данных.

В настоящее время эта возможность не реализована.

3.2 Оформление подписки

Сервис публикации данных не публикует на канале все записи, какие есть. На них нужно подписаться.
Если на запись какого-то типа есть хотя бы один подписчик, то изменения записей этого типа будут публиковаться на соответствующем канале.
Для оформления подписки нужно отправить запрос уже сервису публикации данных, канал которого `#Data.Query`.
Например, для подписки на `ObjectEntity` запрос к сервису может выглядеть так:

```
{
  "Type": "ObjectEntity"
}
```

Запрос к роутеру:

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
  "Payload":
  {
    "Type": "ObjectEntity"
  }
```

```
}
```

Ответ сервисом отправляется в виде такого запроса к роутеру, который доставляет его инициатору запроса:

```
{
  "Command": "response",
  "Channel": "#Data.Query",
  "Id": "0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
  "Payload":
  {
    "Id": "123456",
    "Type": "ObjectEntity",
    "Expires": "2024-07-22T18:25:43.511Z"
  }
}
```

Идентификатор ответа соответствует идентификатору запроса, канал соответствует сервису публикации данных, команда — `response`.

Далее в примерах будут приводиться не запросы к роутеру целиком, а только сами запросы к сервисам и ответы на них, содержащиеся в поле `Payload` запроса `RouterRequest`.

3.3 `SubscribeRequest`

Управление подпиской на данные. Полный набор полей запроса выглядит так (Payload):

```
{
  "Type": "string: Тип данных",
  "Keys": ["int64: Идентификаторы записей"],
  "Id": "int64?: Идентификатор подписки",
  "Subscribe": "bool?: Подписаться -- True, отписаться -- False",
  "Unsubscribe": "bool?: Подписаться -- False, отписаться -- True",
  "Init": "bool?: True -- требуются ли уже имеющиеся записи"
}
```

`"Type"` указывает тип данных записей.

`"Keys"` указывается для тех типов данных, для которых недоступна подписка на весь имеющийся объём.

Это записи, отражающие ход торговли, поэтому подписываться на них можно, только указывая один или несколько ключей, которые соответствуют финансовым инструментам.
В разделе 7. Информационная модель данных биржи для каждого типа данных указано, какой способ подписки следует использовать – по всей таблице или по отдельным ключам.

Идентификатор подписки `"Id"` указывается во всех запросах, кроме первого, когда подписка создаётся.

Флаг `"Subscribe"` определяет требуемое действие с подпиской, значение по-умолчанию True — создание/изменение, False — отписка.
Отписавшись, подписку нельзя снова "переподписать", нужно создавать новую.

Флаг `"Unsubscribe"` является противоположным `"Subscribe"` и может использоваться для улучшения читабельности запросов на подписку и отписку.

Если указать флаг `"Init": "True"`, то в ответ будут присланы все имеющиеся в хранилище данных записи.
Значение по-умолчанию — `False`.

3.4 SubscribeResponse

Ответ на запросы управления подписками. Полный набор полей ответа выглядит так:

```
{
  "Type": "string: Тип данных",
  "Id": "int64: Идентификатор подписки",
  "Subscribed": "bool: Активна ли подписка",
  "Data": [{}],
  "Expires": "DateTime: Дата и время истечения срока подписки",
  "Error":
  {
    "Code": "int: Код ошибки",
    "Message": "string?: Текст сообщения"
  }
}
```

`"Type"` указывает тип данных записей.

Идентификатор подписки `"Id"` нужен для последующих операций с подпиской.

Признак `"Subscribed"` говорит о том, активна ли подписка, или произошла отписка.
Отписка может произойти как в ответ на запрос на отписку, так и при истечении срока жизни подписки (экспирации).

Массив `"Data"` содержит записи, уже имеющиеся в хранилище на момент подписки или добавления к ней новых ключей.
В случае добавления новых ключей к подписке, в этом массиве будут только записи по новым ключам.

Дата и время, указанные в `"Expires"`, говорят о том, когда подписка будет автоматически отписана. Это предусмотрено на случай некорректного завершения работы клиентов во избежание утечек подписок и избыточной публикации данных.

Для продления подписки нужно до наступления времени, указанного в `"Expires"`, отправить новый запрос на подписку, например:

```
{
  "Id": "123456"
}
```

Для продления подписки достаточно указать только её идентификатор, и сервис пришлёт ответ с новым временем экспирации.

Если при обработке запроса произошла ошибка, её код и описание будут в поле `"Error"`.

Помимо продления подписки, для подписки на данные по ключам доступны операции добавления и удаления ключей.
Для добавления ключей следует отправить запрос вида:

```
{
  "Id": "123456",
  "Keys": ["123", "456"]
}
```

Если в подписке этих ключей (123 и 456) не было, они будут добавлены.

Для удаления ключей из подписки, запрос будет выглядеть так:

```
{
  "Id": "123456",
  "Keys": ["123", "456"],
  "Unsubscribe": "True"
}
```

Если из подписки удалить все ключи, подписка будет оставаться активной и может продлеваться, допуская добавление новых ключей в будущем.

4. Монитор состояния терминала

Монитор состояния публикует обновления состояния на канале `#ConnectionState.Bus`:

```
{
  "States":
  {
    "User":
    {
      "Login": "string: Логин пользователя",
      "FullName": "string: ФИО пользователя",
      "AuthStatus": "int: статус соединения"
    },
    "Server":
    {
      "ServerTimeOffset": "time: отставание от серверного времени",
      "ServerTimeZone": {
        "StandardName": "Russia TZ 2 Standard Time"
      }
    },
    "SignService":
    {
      "ReadyToSign": "bool: Готовность ЭЦП"
    }
  }
}
```

Статусы соединения `"AuthStatus"`:

- 0 — оффлайн
- 1 — авторизация в процессе
- 2 — авторизация пройдена
- 3 — ошибка авторизации

Если статус отличается от `2`, работать с терминалом не получится, нужно дожидаться оповещения о возобновлении соединения с этим статусом.

`"ServerTimeOffset"` — сколько нужно прибавить к системному времени (UTC), чтобы получить время сервера.

`"ServerTimeZone.StandardName"` — стандартное наименование часового пояса сервера (всегда МСК).

Терминал выполняет автоматическое переподключение при необходимости (из состояния 1 в 2).

Для отправки торговых поручений, необходимо, чтобы `"SignService.ReadyToSign"` был `True`.

Для получения состояния терминала нужно:

1. Отправить запрос к роутеру:

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
  "Payload":
  {
     "Init" = "True",
     "Subscribe" = true
  }
}
```

2. Отправить запрос на прослушивание шины:

```
{
  "Command": "listen",
  "Channel": "#ConnectionState.Bus"
}
```

3. В получаемом ответе проверять параметры `"AuthStatus"` (должен быть = 2), `"SignService.ReadyToSign"` (должен быть true).

5. Лимит-сервис

Служит для определения доступного лимита (доступного количества) по инструменту перед выставлением заявки.

5.1 LimitRequest

Запрос отправляется в канал сервиса `#Order.Limit.Query`:

```
{
   "Command":"request",
   "Channel":"#Order.Limit.Query",
   "Id":"0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
   "Payload":{
      "IdAccount":"int: Идентификатор счёта, взять из ClientAccountEntity.IdAccount",
      "IdRazdel":"int: Идентификатор портфеля, взять из
SubAccountRazdelEntity.IdRazdel, где IdAccount = указанному выше, IdRazdelGroup -
подходящий рынок",
      "IdObject":"int: Идентификатор выпуска, взять из AssetInfoEntity.IdObject, где
Ticker = тикер инструмента",
      "IdMarketBoard":"int: Идентификатор рынка, взять из
AssetInfoEntity.IdMarketBoard, где IdMarketBoard = нужный рынок",
      "IdDocumentType":"int, взять из AllowedOrderParamsEntity.IdDocumentType",
      "BuySell":"int: Направление сделки - покупка (1) или продажа (-1)",
      "Price":"double: Цена",
      "IdOrderType:":"int: Вид заявки - рыночная (1) или лимитная (2)",
      "LimitRequestType":"int: Вид запрашиваемого лимита - исходя из количества
свободных денег (3) или исходя из стоимости портфеля (4)"
   }
}
```

5.2 LimitResponse

Ответ лимит-сервиса имеет следующий вид:

```
{
  "Quantity": "int: Доступное количество",
  "QuantityForOwnAssets": "int: Доступное количество без плеча"
}
```

6. Торговый сервис

6.1 Выставление заявки

Количество передается в штуках. Можно выставить минимальный объем равный количеству в одном лоте.

Запрос отправляется на канал `#Order.Enter.Query`:

```
{
   "Command":"request",
   "Channel":"#Order.Enter.Query",
   "Id":"0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
   "Payload":{
      "IdAccount":"int: Идентификатор счёта, взять из ClientAccountEntity.IdAccount",
      "IdSubAccount":"int: Идентификатор субсчёта, взять из
ClientSubAccountEntity.IdSubAccount, где IdAccount = указанному выше",
      "IdRazdel":"int: Идентификатор портфеля, взять из
SubAccountRazdelEntity.IdRazdel, где IdAccount = указанному выше, IdRazdelGroup -
подходящий рынок",
      "IdPriceControlType":"3",
      "IdObject":"int: Идентификатор выпуска, взять из AssetInfoEntity.IdObject, где
Ticker = тикер инструмента",
      "LimitPrice":"double: Лимитная цена",
      "StopPrice":"double: Стоп цена",
      "LimitLevelAlternative":"double: Альтернативная лимитная цена",
      "BuySell":"int: Направление - купить (1) или продать (-1)",
      "Quantity":"int: Количество в штуках",
      "Comment":"string: Комментарий",
      "IdAllowedOrderParams":"int: Идентификатор комбинации параметров, взять из
AllowedOrderParamsEntity.IdAllowedOrderParams"
   }
}
```

`IdPriceControlType` может принимать значения:

- 1 - контролировать условие по значению Индекса.
- 2 - контролировать условие по транслируемому биржей значению расчетной рыночной котировки.
- 3 - контролировать условие по цене сделок в течение торговой сессии (без учета сделок периода открытия и закрытия).
- 4 - контролировать условие по цене сделок в течение основной торговой сессии (без учета сделок дополнительных вечерней и/или утренней сессий периода открытия и закрытия).
- 5 - контролировать условие по доходности сделок в течение торговой сессии (без учета сделок периода открытия и закрытия).
- 6 - контролировать условие по доходности сделок в течение основной торговой сессии (без учета сделок дополнительных вечерней и/или утренней сессий периода открытия и закрытия).
- 7 - контролировать условие по цене спроса в течение торговой сессии (без учета ордеров периода открытия и закрытия).
- 8 - контролировать условие по цене спроса в течение основной торговой сессии (без учета дополнительных вечерней и/или утренней сессий периода открытия и закрытия).
- 9 - контролировать условие по доходности спроса в течение торговой сессии (без учета периода открытия и закрытия).

- 10 - контролировать условие по цене спроса в течение основной торговой сессии (без учета дополнительных вечерней и/или утренней сессий периода открытия и закрытия).
- 11 - контролировать условие по цене предложения в течение торговой сессии (без учета периода открытия и закрытия).
- 12 - контролировать условие по цене предложения в течение основной торговой сессии (без учета дополнительных вечерней и/или утренней сессий периода открытия и закрытия).
- 13 - контролировать условие по доходности предложения в течение торговой сессии (без учета периода открытия и закрытия).
- 14 - контролировать условие по цене предложения в течение основной торговой сессии (без учета дополнительных вечерней и/или утренней сессий периода открытия и закрытия).

Заполнение Order.Enter.Query для разных типов заявок:

Описание	ClientEnterOrderReq			AllowedOrderParams
	LimitPrice	StopPrice	LimitLevelAlternative	IdOrderType
Рыночная заявка				MKT = 1
Лимитная заявка	цена			LMT = 2
Стоп-маркет		стоп-цена		STP = 7
Стоп-лимит		стоп-цена	цена	STL = 8
Трейлинг-лимит	цена	старт-цена		TRL = 9
Трейлинг-стоп-маркет	старт-цена	цена		TRS = 10
Трейлинг-стоп-лимит	старт-цена	стоп-цена	цена	TSL = 11
Стоп-маркет+тейк-профит	цена-профит	стоп-цена		RS = 12
Стоп-лимит+тейк-профит	цена-профит	стоп-цена	цена	BSL = 13
Трейлинг стоп-маркет+тейк-профит	цена-профит	стоп-цена	старт-цена	TBRS = 28

Цена: цена, по которой хотим исполнения заявки.

Стоп-цена: цена, по которой активируется исполнение стоп-заявки.

Старт-цена: рыночная цена, от которой начинается трейлинг.

Цена-профит: цена, активирующая тейк-профит.

Ответ:

```
{
  "ResponseStatus": "int: Статус ответа: 0 -- ОК, другое -- ошибка",
  "Message": "string? Сообщение к статусу",
  "Error":
  {
    "Code": "int: Код ошибки",
    "Message": "string: Сообщение об ошибке"
```

```
  },
  "Value":
  {
    "ClientOrderNum": "int: Клиентский номер заявки",
    "NumEDocument": "int64: Брокерский идентификатор заявки",
    "ErrorCode": "int: Код ошибки для терминала, 0 - ОК",
    "ErrorText": "string: Тест ошибки для терминала"
  }
}
```

6.2 Снятие заявки

Запрос отправляется на канал `#Order.Cancel.Query`:

```
{
   "Command":"request",
   "Channel":"#Order.Cancel.Query",
   "Id":"0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
   "Payload":{
      "IdAccount":"int: Идентификатор счёта",
      "IdSubAccount":"int: Идентификатор субсчёта",
      "IdRazdel":"int: Идентификатор портфеля",
      "NumEDocumentBase ":"int: Брокерский идентификатор заявки, взять из ответа на
выставление заявки, NumEDocument"
   }
}
```

Ответ:

```
{
  "ResponseStatus": "int: Статус ответа: 0 -- ОК, другое -- ошибка",
  "Message": "string? Сообщение к статусу",
  "Error":
  {
    "Code": "int: Код ошибки",
    "Message": "string: Сообщение об ошибке"
  },
  "Value":
  {
    "ClientOrderNum": "int: Клиентский номер заявки",
    "NumEDocument": "int64: Брокерский идентификатор заявки",
```

```
    "ErrorCode": "int: Код ошибки для терминала, 0 - ОК",
    "ErrorText": "string: Тест ошибки для терминала"
  }
}
```

7. Информационная модель данных биржи

Далее описываются сущности (entity), которые хранятся во внутреннем хранилище терминала и доступны к подписке через сервис публикации данных.
Первым полем всегда идёт идентификатор (первичный ключ).
Поля, начинающиеся с `Id` являются вторичными ключами, ссылающимися на другие сущности, поэтому в их описании тип данных (int64) и слово "идентификатор" опускаются.

AssetInfoEntity

Информация о выпуске и его финансовых инструментах.

```
{
  "IdObject": "int: Идентификатор выпуска",
  "Ticker": "string: Биржевой тикер",
  "ISIN": "string: Международный идентификатор",
  "Name": "string: Наименование",
  "Description": "string: Описание",
  "Nominal": "double: Номинальная стоимость",
  "IdObjectType": "Тип выпуска",
  "IdObjectGroup": "Группа выпусков",
  "IdObjectBase": "Базовый выпуск",
  "IdObjectFaceUnit": "Выпуск валюты номинала",
  "MatDateObject": "date?: Дата экспирации",
  "Instruments":  [
    {
      "IdFi": "int: Идентификатор фининструмента",
      "RCode": "string: Код портфеля",
      "IsLiquid": "bool: Ликвидность",
      "IdMarketBoard": "Рынок"
    }
  ]
}
```

Для торговли, например, акциями Сбербанка, нужно найти соответствующий выпуск (по тикеру или ISIN), и в нём подходящий фининструмент.
Если выпуск торгуется на нескольких рынках, то один из них будет иметь флаг `IsLiquid` – рекомендуется торговать именно таким фининструментом.
Флаг `IsLiquid` может быть у нескольких фининструментов, по одному на биржу.
Фининструмент с признаком `IsLiquid` на приоритетной бирже идёт первым.

Другие сущности, на которые ссылается AssetInfo:

ObjectTypeEntity

Тип выпуска.

```
{
  "IdObjectType": "int: Идентификатор типа выпуска",
  "IdObjectGroup": "Группа выпусков",
  "CodeObjectType": "string: Код типа",
  "NameObjectType": "string: Имя типа",
  "ShortNameObjectType": "string: Краткое имя типа"
}
```

ObjectGroupEntity

Группа выпусков:

```
{
  "IdObjectGroup": "int: Идентификатор группы",
  "NameObjectGroup": "string: Имя группы"
}
```

MarketBoardEntity

Рынок.

```
{
  "IdMarketBoard": "int: Идентификатор рынка",
  "NameMarketBoard": "string: Имя рынка",
  "DescMarketBoard": "string: Описание рынка",
  "RCode": "string: Код портфеля, который торгуется на рынке",
  "IdObjectCurrency": "Выпуск валюты торговли рынка"
}
```

Для получения информации о параметрах инструмента и ходе торгов потребуются следующие сущности (ниже), подписаться на которые можно только по ключу (`IdFi`):

FinInfoParamsEntity

Параметры торговли, обычно не изменяются во время сессии. Можно загружать только по конкретному инструменту.
Есть ограничение на количество инструментов, на которые можно подписаться одновременно - 500 (суммарно по FinInfoParamsEntity + FinInfoLastEntity).

```
{
  "IdFi": "Идентификатор фининструмента",
  "IdSession": "Идентификатор сессии",
  "IdTradePeriodStatus": "Статус торговли, 6 - торговля",
  "SessionDate": "date: Дата сессии",
  "Lot": "int: Штук в лоте",
  "PriceStep": "double: Шаг цены",
  "PriceStepCost": "double: Стоимость шага цены",
  "IdObjectCurrency": "Выпуск валюты цены",
  "PSTNKD": "double: НКД",
  "UpPrice": "double: Максимальная цена",
  "DownPrice": "double: Минимальная цена",
  "GtBuy": "double: Гарантийное обеспечение на покупку",
  "GtSell": "double: Гарантийное обеспечение на продажу",
  "FaceValue": "double: Номинал"
}
```

FinInfoLastEntity

Состояние торговли, меняющееся с каждой сделкой на бирже.

```
{
  "IdFi": "Идентификатор фининструмента",
  "IdSession": "Идентификатор сессии",
  "IdTradePeriodStatus": "Статус торговли, 6 - торговля",
  "Last": "double: Цена последней сделки",
  "PrevLast": "double: Цена последней сделки предыдущей сессии",
  "LastQty": "int: Количество последней сделки",
  "LastTime": "datetime: Дата и время последней сделки",
  "Yield": "double: Доходность последней сделки",
```

```
  "Open": "double: Цена первой сделки сессии",
  "High": "double: Максимальная цена в сессии",
  "Low": "double: Минимальная цена в сессии",
  "WaPrice": "double: Средневзвешенная по объёму цена в сессии",
  "YieldWaPrice": "double: Средневзвешенная по объёму доходность",
  "NumTrades": "double: Число сделок за сессию",
  "VolToday": "double: Объём за сессию в штуках",
  "ValToday": "double: Объём за сессию в валюте торгов"
}
```

FinInfoOrderBookEntity

Состояние торговли, меняющееся с каждой выставленной или снятой заявкой в биржевом торговом стакане.

```
{
  "IdFi": "Идентификатор фининструмента",
  "IdSession": "Идентификатор сессии",
  "IdTradePeriodStatus": "Статус торговли, 6 - торговля",
  "Bid": "double: Лучший бид",
  "Ask": "double: Лучший оффер",
  "BidQty": "double: Количество лотов по цене лучшего бида",
  "AskQty": "double: Количество лотов по цене лучшего оффера",
  "SumBid": "double: Суммарное количество лотов всех бидов",
  "SumAsk": "double: Суммарное количество лотов всех офферов",
  "NumBids": "int: Количество бидов",
  "NumAsks": "int: Количество офферов",
  "HighBid": "double: Максимальный бид за сессию",
  "LowAsk": "double: Минимальный оффер за сессию"
}
```

OrderBookEntity

Биржевой стакан, состоит из строк, одна строка – одна цена.

Обновляется в реальном времени. Глубина - 40 уровней. Одновременно можно запросить не более 30 стаканов.

```
{
  "IdFi": "Идентификатор фининструмента",
```

```
  "Lines":
  [
    {
      "Price": "double: Цена",
      "BuyQty": "int: Количество лотов на покупку",
      "SellQty": "int: Количество лотов на продажу"
    }
  ]
}
```

AllTradeEntity

Лента сделок.

Одновременно можно запросить не более 100 лент.

```
{
    "IdFI":"int: Идентификатор фининструмента",
    "TradeNo":"long: Номер сделки",
    "IdTradeType":"int: Номер типа сделки по справочнику (1 - TRD Сделка
купли-продажи)",
    "TradeTime":"datetime: Дата и время регистрации сделки (время биржи)",
    "Price": "double: Цена",
    "BuySell":"int: Направление сделки - покупка (1) или продажа (-1)",
    "Yield":"double: Доходность последней сделки",
    "Qty":"double: Количество штук",
    "Value":"double: Объем сделки, выраженный в руб",
    "Pos":"int: Объем открытых позиций, контрактов по итогам сделки",
    "BaTime":"datetime: Дата и время изменения  обновления  записи по таймеру
Биржевого Агента"
}
```

Архивные данные (котировки)

1. Запрос архивных данных
Запрос отправляется на канал `#Archive.Query`:

```
{
   "Command":"request",
```

```
   "Channel":"#Archive.Query",
   "Id":"0407FC6C-4F9A-40CC-A26D-F155FAA09BEB",
   "Payload":{
      "IdFi": "int: Идентификатор фининструмента",
      "CandleType": "int: Вид свечи, стандартная OHLCV - 0, или MPV - 2",
      "Interval": "string: Таймфрейм (second, minute, hour, day, week, month)",
      "Period": "int: Множитель таймфрейма",
      "TimeFrame": "string: Таймфрейм. Если указан, Interval и Period, то не
используются",
      "FirstDay": "DateTime: Первый запрашиваемый день",
      "DaysCount": "int: Количество дней с данными, которые
запрашиваются.Положительное значение от FirstDay в будущее, отрицательное - в
прошлое. Данные за FirstDay возвращаются при любом направлении запроса, если это
торговый день",
      "LastDay": "DateTime: Последний запрашиваемый день.Если DaysCount не указан,
запрашиваются данные за дни между FirstDay и LastDay",
      "AtLeastOneCandle": "bool: При остутствии свечей, должна быть возвращена хотя
бы одна",
      "TakeLastNCandles": "int: Если значение больше нуля, то вернуть из отобранных
свечей только последние",
      "Scope": "string: Любой идентификатор. Если указан, то уже выполняющийся
запрос с таким же идентификатором будет отменён"
   }
}
```

2. Трансляция архивных данных

Транслируются данные за текущий день (получение текущей свечи в реальном времени). Оформляется подписка на канал (только listen), включающий в название инструмент, таймфрейм и тип свечей.

Например: #Archive.Bus.OHLCV.144950.H1 означает подписку на стандартные свечи, таймфрейма 1 час (1 - множитель), инструмента IdFi=144950.

```
{
  "Command": "listen",
  "Channel": "#Archive.Bus.OHLCV.144950.H1"
}
```

Коды таймфреймов:

- L - милисекунда
- S - секунда
- M - минута

- H - час
- D - день
- W - неделя
- N - месяц
- Y - год

3. Ответ
Ответ для OHLCV свечей:

```
{
    "Open": "double: Цена открытия",
    "Close": "double: Цена закрытия",
    "Low": "double: Минимальная цена за торговую сессию",
    "High": "double: Максимальная цена за торговую сессию",
    "Volume": "long: Объем на уровне цены по таймфрейму",
    "VolumeAsk": "long: Объем на продажу на уровне цены по таймфрейму",
    "OpenInt": "long: Открытый интерес (для фьючерсов)",
    "Time": "DateTime: Дата и время"
}
```

Ответ для MPV свечей:

```
{
    "Open": "double: Цена открытия",
    "Close": "double: Цена закрытия",
    "Time": "DateTime: Дата и время",
    "Levels": "MPVLevel[]: Уровни цен"
}

MPVLevel:
{
    "Price": "double: Цена",
    "Volume": "long: Объем на уровне цены по таймфрейму",
    "VolumeAsk": "long: Объем на продажу на уровне цены по таймфрейму",
}
```

8. Информационная модель данных клиента

Сущности (entity), описывающие состояние счетов клиента, также хранятся во внутреннем хранилище терминала и доступны к подписке через сервис публикации данных.

Первым полем всегда идёт идентификатор (первичный ключ).
Поля, начинающиеся с `Id` являются вторичными ключами, ссылающимися на другие сущности, поэтому в их описании тип данных (int64) и слово "идентификатор" опускаются.

ClientAccountEntity

Счёт клиента.

```
{
  "IdAccount": "int: Идентификатор счёта"
}
```

ClientSubAccountEntity

Субсчёт клиента.

```
{
  "IdSubAccount": "int: Идентификатор субсчёта",
  "IdAccount": "Счёт клиента"
}
```

SubAccountRazdelEntity

Портфель субсчёта клиента.

```
{
  "IdRazdel": "int: Идентификатор портфеля",
  "IdAccount": "Счёт клиента",
  "IdSubAccount": "Субсчёт клиента",
  "IdRazdelGroup": "int: Группа портфелей",
  "RCode": "string: Код раздела"
}
```

Группы портфелей:

- 1 — РЦБ (рынок ценных бумаг)
- 2 — ФОРТС
- 3 — ВР (валютный рынок)
- 4 — НТР (неторгуемый раздел)

ClientPositionEntity

Текущая позиция по выпуску, данные по текущей торговой сессии.

```
{
  "IdPosition": "int: Идентификатор позиции",
  "IdAccount": "Счёт",
  "IdSubAccount": "Субсчёт",
  "IdRazdel": "Портфель",
  "IdObject": "Выпуск",
  "IdFiBalance": "Фининструмент для расчёта стоимости позиции",
  "IdBalanceGroup": "Группа портфелей (см. SubAccountRazdel)",
  "AssetsPercent": "double: Процентная доля позиции в субсчёте",
  "PSTNKD": "double: НКД",
  "IsMoney": "bool: Деньги -- валютная позиция",
  "IsRur": "bool: Деньги -- валютная, рублёвая позиция",
  "UchPrice": "double: Учётная цена",
  "TorgPos": "double: Текущая позиция",
  "Price": "double: Текущая цена",
  "DailyPL": "double: Текущая прибыль/убыток (ПУдн)",
  "DailyPLPercentToMarketCurPrice": "double: ПУдн%",
  "BackPos": "double: Входящая позиция",
  "PrevQuote": "double: Входящая цена (закрытия предыдущей сессии)",
  "TrnIn": "double: Объём внешних зачислений",
  "TrnOut": "double: Объём внешних списаний",
  "DailyBuyVolume": "double: Объём покупок за сессию",
  "DailySellVolume": "double: Объём продаж за сессию",
  "DailyBuyQuantity": "double: Количество покупок за сессию",
  "DailySellQuantity": "double: Количество продаж за сессию",
  "NKD": "double: НКД (накопленный купонный доход)",
  "PriceStep": "double: Шаг цены",
  "Lot": "int: Размер в лотах",
  "NPLtoMarketCurPrice": "double: Номинальная прибыль/убыток (НПУ)",
  "NPLPercent": "double: НПУ%",
  "PlanLong": "double: Плановая длинная позиция",
  "PlanShort": "double: Плановая короткая позиция"
}
```

Плановая позиция учитывает текущую позицию плюс неисполненные остатки заявок на покупку и продажу.

ClientBalanceEntity

Баланс портфеля субсчёта. Идентификатор (первичный ключ) вычисляется по формуле: `DataId=IdSubAccount * 8 + IdRazdelGroup`.
Для каждого субсчёта может быть максимум четыре баланса: РЦБ (1), ФОРТС (2), ВР (3) и НТР (4).

```
{
  "DataId": "int64: Идентификатор баланса",
  "IdAccount": "Счёт клиента",
  "IdSubAccount": "Субсчёт клиента",
  "IdRazdelGroup": "int: Группа портфелей",
  "MarginInitial": "double: Начальная маржа",
  "MarginMinimum": "double: Минимальная маржа",
  "MarginRequirement": "double: Маржинальные требования",
  "Money": "double: Рубли",
  "MoneyInitial": "double: Входящие рубли (на начало сессии)",
  "Balance": "double: Баланс",
  "PrevBalance": "double: Входящий баланс",
  "PortfolioCost": "double: Стоимость портфеля",
  "LiquidBalance": "double: Ликвидная стоимость портфеля",
  "Requirements": "double: Требования",
  "ImmediateRequirements": "double: Безотлагательные требования",
  "NPL": "double: Номинальная прибыль/убыток (НПУ)",
  "DailyPL": "double: Дневная прибыль/убыток (ПУдн)",
  "NPLPercent": "double: НПУ%",
  "DailyPLPercent": "double: ПУдн%",
  "NKD": "double: Накопленный купонный доход (НКД)"
}
```

OrderEntity

Заявки доступны за текущую сессию.

Сопоставить с выставленной заявкой можно по ClientOrderNum или NumEDocument (приходит в ответе на выставление).

```
{
  "NumEDocument": "int64: Идентификатор заявки",
  "ClientOrderNum": "int: Клиентский номер заявки",
  "IdAccount": "Счёт клиента",
  "IdSubAccount": "Субсчёт клиента",
  "IdRazdel": "Портфель субсчёта клиента",
```

```
  "IdAllowedOrderParams": "int: Идентификатор комбинации параметров",
  "AcceptTime": "datetime: Дата приёма заявки",
  "IdOrderType:": "int: Вид заявки - рыночная (1) или лимитная (2)",
  "IdObject": "Выпуск",
  "IdMarketBoard": "Рынок",
  "LimitPrice": "double: Цена лимит-ордера",
  "BuySell": "int: Направление сделки - покупка (1) или продажа (-1)",
  "Quantity": "int: Количество, шт.",
  "Comment": "string: Комментарий",
  "Login": "string: Логин инициатора",
  "IdOrderStatus": "int: Статус заявки",
  "Rest": "int: Неисполненный остаток",
  "Price": "double: Цена",
  "BrokerComment": "string: Комментарий брокера"
}
```

Статусы заявок:

- 5 HiddenTime — ожидает активации по времени
- 6 HiddenOrder — ожидает активации по другой заявке
- 10 Accepted — принят брокером
- 12 HiddenPeriod — ожидает наступления периода торгов
- 14 Working — активный
- 15 Kill — помечен для удаления
- 18 Rejected — отклонен брокером
- 20 Filled — исполнен
- 21 Cancelled — отменён
- 23 Edit — помечен к удалению для редактирования
- 24 Moving — редактируется
- 25 ExchangeRejected — удалён с биржи до попадания в стакан
- 26 ExchangeRemoved — удалён с биржи после попадания в стакан

OrderRevisionEntity

История состояния заявок (последовательность жизни заявки).

```
{
  "Quantity":"int: Количество",
  "IdOrderType:": "int: Вид заявки - рыночная (1) или лимитная (2)",
  "Price": "double: Цена",
  "Rest": "int: Неисполненный остаток",
  "FilledPrice": "double: Средневзвешенная цена",
  "LimitPrice": "double: Цена лимит-ордера",
```

```
  "StopPrice": "double: Цена стоп-ордера",
  "IdOrderStatus": "int: Статус заявки",
  "IdPriceType": "int: Тип цены заявки",
  "IdQuantityType": "int: Тип измерения количества",
  "WithDrawTime": "datetime: Дата истечения срока действия ордера",
  "ChangeTime": "datetime: Дата и время регистрации биржевого номера ордера",
  "IdLifeTime": "int: Срок действия торговой заявки",
  "OpenQuantity": "int: Открытое количество",
  "BrokerComment": "string: Комментарий брокера",
  "BrokerErrorCode": "string: Ошибка брокера",
  "OrderStatusRevision": "int: Последовательность состояний заявки, меньше -
произошло раньше"
}
```

ClientOperationEntity

Операции (сделки).

```
{
  "IdOperation": "Идентификатор операции",
  "TimeOperation": "datetime: Дата/время операции",
  "IdMarketBoard": "Рынок",
  "IdOperationType": "TRD",
  "IdObject": "Выпуск",
  "BuySell": "int: Направление сделки - покупка (1) или продажа (-1)",
  "Quantity": "int: Количество, шт.",
  "Price": "double: Цена",
  "IdAccount": "Счёт клиента",
  "IdSubAccount": "Субсчёт клиента",
  "IdRazdel": "Портфель субсчёта клиента",
  "NumEDocument": "int64: Идентификатор заявки"
}
```

AllowedOrderParamsEntity

Комбинация параметров заявки.
Справочник содержит все доступные комбинации, а при отправке заявки используется только идентификатор комбинации `IdAllowedOrderParams`.
При выставлении заявки по фининструменту, нужно найти такую запись, чтобы в ней были значения группы выпусков и рынка, соответствующие инструменту, а вид заявки соответствовал требуемой (рыночная или лимитная).

```
{
  "IdAllowedOrderParams": "int: Идентификатор комбинации",
  "IdObjectGroup": "Группа выпусков",
  "IdMarketBoard": "Рынок",
  "IdOrderType:": "int: Вид заявки - рыночная (1) или лимитная (2)",
  "IdDocumentType": "1",
  "IdQuantityType": "1",
  "IdPriceType": "1",
  "IdLifeTime": "9",
  "ExecutionType": "1"
}
```

9. Примеры

9.1 Стартовые операции

Перед началом работы с сервисами необходимо:

1. Запустить Терминал, установить сокетное соединение.
2. Проверить состояние терминала (подробнее в п.4. Монитор состояния терминала). Если SignService.ReadyToSign = false, то выпустить сертификат ЭП в терминале (Меню->Личный кабинет->Электронная подпись).

9.2 Поиск инструмента для торговли

Допустим, известен тикер акций Сбербанка — "SBER".
Нужно найти его запись `AssetInfoEntity`.
Для этого отправляются запросы на прослушивание шины `"#Data.Bus.AssetInfoEntity"` и запрос на подписку на этот справочник:

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.AssetInfoEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
```

```
  "Id": "1",
  "Payload":
  {
    "Type": "AssetInfoEntity",
    "Init": "True"
  }
}
```

В ответ приходит подписка и имеющиеся данные, ниже приведена только одна искомая запись, с полем Ticker="SBER":

```
{
  "Command": "response",
  "Channel": "#Data.Query",
  "Id": "1",
  "Payload":
  {
    "Id": "1001",
    "Type": "AssetInfoEntity",
    "Expires": "2024-07-23T10:25:11.210Z",
    "Data": [
      {
        "IdObject": 285270,
        "Ticker": "SBER",
        "ISIN": "RU0009029540",
        "Name": "Сбербанк",
        "Description": "Акции обыкновенные ПАО Сбербанк, Рег№ 10301481B",
        "Nominal": 3.0,
        "IdObjectType": 1,
        "IdObjectGroup": 1,
        "IdObjectBase": 825227,
        "IdObjectFaceUnit": 174368,
        "Instruments": [
          {
            "IdFi": 144950,
            "RCode": "MICEX",
            "IsLiquid": true,
            "IdMarketBoard": 92
          }
        ]
      }
    ]
```

```
  }
}
```

Отсюда можно извлечь следующую информацию, которая потребуется далее:

- IdObject: 295270 — идентификатор выпуска
- IdObjectGroup: 1 – группа выпусков (Stocks)
- IdObjectFaceUnit: 174368 — валюта RUB, можно найти в том же справочнике
- IdFi: 144950 — идентификатор ликвидного фининструмента
- IdMarketBoard: 92 — идентификатор рынка

9.3 Работа с заявками

Описан типовой процесс работы с заявками:

1. Произвести поиск инструмента (п. Поиск инструмента для торговли). Для выставления заявки будет использован параметр IdObject.
2. Запросить счёт - IdAccount, подписаться на ClientAccountEntity.
3. Запросить субсёт - IdSubAccount, подписаться на ClientSubAccountEntity. У одного счёта один субсчёт.
4. Запросить идентификатор портфеля - IdRazdel, подписаться на SubAccountRazdelEntity. Должен совпадать RCode инструмента (AssetInfoEntity) и раздела (SubAccountRazdelEntity).
5. Запросить комбинацию параметров заявки, подписаться на AllowedOrderParamsEntity. Для выставления заявки будет использован параметр IdAllowedOrderParams.
6. Проверить доступные лимиты (подробнее в п.5. Лимит-сервис). Если указать в заявке большее количество чем доступно, то такая заявка будет отклонена.
7. Выставить заявку (подробнее в п.6.1 Выставление заявки).
8. Запросить список заявок (подписаться на OrderEntity).
9. Снять заявку (подробнее в п.6.2 Снятие заявки).
10. Если нужно изменить заявку, то снимается текущая и выставляется новая.

Подписка на счета, субсчета, портфели, заявки

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.ClientAccountEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
```

```
  "Id": "1",
  "Payload":
  {
    "Type": "ClientAccountEntity",
    "Init": "True"
  }
}
```

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.ClientSubAccountEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "2",
  "Payload":
  {
    "Type": "ClientSubAccountEntity",
    "Init": "True"
  }
}
```

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.SubAccountRazdelEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "3",
  "Payload":
  {
    "Type": "SubAccountRazdelEntity",
    "Init": "True"
  }
}
```

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.OrderEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "4",
  "Payload":
  {
    "Type": "OrderEntity",
    "Init": "True"
  }
}
```

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.OrderRevisionEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "5",
  "Payload":
  {
    "Type": "OrderRevisionEntity",
    "Init": "True"
  }
}
```

Поиск комбинации параметров заявки

Выполняется подписка на справочник комбинаций:

```
{
  "Command": "request",
  "Channel": "#Data.Query",
```

```
  "Id": "9",
  "Payload":
  {
    "Type": "AllowedOrderParamsEntity",
    "Init": "True"
  }
}
```

В ответе, среди записей, нужно найти параметры, соответствующие инструменту и параметрам заявки:

```
{
  "IdAllowedOrderParams": 15,
  "IdObjectGroup": 1,
  "IdMarketBoard": 92,
  "IdDocumentType": 1,
  "IdOrderType": 1,
  "IdQuantityType": 1,
  "IdPriceType": 1,
  "IdLifeTime": 9,
  "IdExecutionType": 1
}
```

Например, для SVU4 получили запись из справочника AssetInfoEntity:

```
{
  IdObject: 831064,
  Ticker: "SVU4",
  ISIN: "SILV-9.24",
  IdObjectType: 17,
  Instruments: [{
    IdFi = 2015326,
    IdMarketBoard: 56,
    RCode: "FORTS")
  }]
}
```

Отсюда нужно взять параметры IdObjectType=17 и IdMarketBoard=56.

У AllowedOrderParamEntity есть 8 параметров:

Фиксированные:

- IdDocumentType - всегда 1 (вид документа - торговый приказ)
- IdQuantityType - всегда 1 (количество в штуках)
- IdPriceType - всегда 1 (тип цены заявки - цена инструмента)
- IdExecutionType - всегда 3 (в торговую систему передаётся как лимитная заявка)

По инструменту:

- IdObjectGroup - должен совпадать с параметром инструмента, в данном случае 17
- IdMarketBoard - должен совпадать с параметром инструмента, в данном случае 56

По заявке:

- IdOrderType - 1 рыночная, 2 лимитная
- IdLifeTime - 5 исполнять до конца дня, 9 - исполнять 30 дней

Для примера рыночной заявки, действующей до конца дня, подойдёт такая запись:

```
{
  IdAllowedOrderParams: 151,
  IdObjectGroup: 17,
  IdMarketBoard: 56,
  IdDocumentType: 1,
  IdOrderType: 1,
  IdQuantityType: 1,
  IdPriceType: 1,
  IdLifeTime: 9,
  IdExecutionType: 3
}
```

Соответственно, при выставлении заявки, нужно указать в запросе IdAllowedOrderParams: 151.

Отправка заявки

На сервис торговли в канал `#Order.Enter.Query` отправляется запрос:

```
{
  "IdAccount": "123456",
  "IdSubAccount": "9999223",
  "IdRazdel": "232323",
  "IdPriceControlType": "3",
  "IdObject": "295270",
  "BuySell": "1",
  "Quantity": "10",
  "Comment": "",
```

```
  "IdAllowedOrderParams": "15"
}
```

В случае успешности выставления заявки ответ будет таким:

```
{
  "ResponseStatus": "0",
  "Value":
  {
    "ClientOrderNum": "12345",
    "NumEDocument": "65654123456",
    "ErrorCode": "0"
  }
}
```

Снятие заявки

На сервис торговли в канал `#Order.Cancel.Query` отправляется запрос:

```
{
  "IdAccount": "123456",
  "IdSubAccount": "9999223",
  "IdRazdel": "232323",
  "NumEDocumentBase": "65654123456"
}
```

В случае успешности выставления заявки ответ будет таким:

```
{
  "ResponseStatus": "0",
  "Value":
  {
    "ClientOrderNum": "12345",
    "NumEDocument": "65654123456",
    "ErrorCode": "0"
  }
}
```

9.4 Подписка на балансы, позиции

Для торговли инструментом нужно знать состояние счёта, для чего отправляются следующие запросы:

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.ClientBalanceEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "2",
  "Payload":
  {
    "Type": "ClientBalanceEntity",
    "Init": "True"
  }
}
```

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.ClientPositionEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "3",
  "Payload":
  {
    "Type": "ClientPositionEntity",
    "Init": "True"
  }
}
```

В ответ приходят подписки на данные. Если открытых позиций нет, то в начале торгов записей не будет, например:

```
{
  "Command": "response",
  "Channel": "#Data.Query",
  "Id": "5",
  "Payload":
  {
    "Id": "1002",
    "Type": "ClientBalanceEntity",
    "Expires": "2024-07-23T10:25:11.210Z"
  }
}
```

9.5 Подписка на торговый стакан (OrderBook)

В этом запросе указывается IdFi (идентификатор фининструмента):

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "8",
  "Payload":
  {
    "Type": "OrderBookEntity",
    "Keys": [144950],
    "Init": "True"
  }
}
```

9.6 Подписка на историю операций

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.ClientOperationEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
```

```
  "Id": "5",
  "Payload":
  {
    "Type": "ClientOperationEntity",
    "Init": "True"
  }
}
```

9.7 Подписка на ленту сделок (AllTradeEntity)

В этом запросе указывается IdFi (идентификатор фининструмента):

```
{
  "Command": "listen",
  "Channel": "#Data.Bus.AllTradeEntity"
}
```

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "8",
  "Payload":
  {
    "Type": "AllTradeEntity",
    "Keys": [144950],
    "Init": "True"
  }
}
```

9.8 Получение архивных данных (котировки, история котировок)

```
{
  "Command": "request",
  "Channel": "#Archive.Query",
  "Id": "6",
  "Payload":
  {
    "IdFi": 144947,
    "IdObject": null,
    "ISIN": null,
```

```
    "Ticker": null,
    "CandleType": 0,
    "Interval": "minute",
    "Period": 5,
    "TimeFrame": null,
    "FirstDay": "2024-03-24T00:00:00",
    "DaysCount": null,
    "LastDay": "2024-08-22T23:59:00+03:00",
    "AtLeastOneCandle": false,
    "TakeLastNCandles": 0,
    "Scope": ""
  }
}
```

```
{
  "Command": "listen",
  "Channel": "#Archive.Bus.OHLCV.144950.H1"
}
```

9.9 Подписка на параметры торговли по инструменту (FinInfoParams)

В этом запросе указывается IdFi (идентификатор фининструмента):

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "6",
  "Payload":
  {
    "Type": "FinInfoParamsEntity",
    "Keys": [144950],
    "Init": "True"
  }
}
```

9.10 Подписка на данные торговли по инструменту (FinInfoLast)

В этом запросе указывается IdFi (идентификатор фининструмента):

```
{
  "Command": "request",
  "Channel": "#Data.Query",
  "Id": "7",
  "Payload":
  {
    "Type": "FinInfoLastEntity",
    "Keys": [144950],
    "Init": "True"
  }
}
```

В этом примере приведены вымышленные идентификаторы счёта, субсчёта, раздела и заявок.