Voice2X

Диалоговая система обеспечения технологического контроля в промышленных акустических условиях с применением технологий искусственного интеллекта

STC-S876
Руководство программиста
ЦВАУ.01128-01 33

Версия документа: 1.0

ЦРТ | ГРУППА КОМПАНИЙ

Содержание

Условные обозначения

Форматирование текста

В руководстве приняты следующие обозначения:

Полужирный — применяется для написания наименований управляющих элементов (кнопки), информационных элементов (заголовки и названия экранов).

Полужирный курсив — используется для написания имён файлов и путей доступа к ним.

Курсив — для описания значений перечислений и элементов.

Оформление материала

Сведения информационного характера: заметки, примеры использования.

Ссылки на дополнительные информационные материалы: паспорта, руководства, инструкции.

Сведения рекомендательного характера.

Важные сведения, указание на действия, которые необходимо выполнить в обязательном порядке.

Сокращения и термины

REST, REST API

Representational **S**tate **T**ransfer и **A**pplication **P**rogramming **I**nterface. Программный интерфейс для обмена данными и состоянием через HTTP-вызовы.

WS

WebSocket. Протокол связи, предназначенный для обмена информацией в режиме реального времени.

АРМ

Автоматизированное рабочее место.

ОС

Операционная система.

МИС, ИС

Сторонняя многопользовательская информационная система.

ЦОД

Центр обработки данных.

Обзор API

Спецификация содержит описание принципов и методов взаимодействия с программным интерфейсом ПО Voice2X.

Что такое Voice2X?

Voice2X представляет собой клиент-серверную диалоговую систему обеспечения технологического контроля в промышленных акустических условиях с применением технологий искусственного интеллекта. В зависимости от используемой специальной речевой модели она может использоваться в разных предметных областях, с успехом распознавая специфические термины.

Использование Voice2X Core API

Voice2X Core API предназначен для нативной интеграции со сторонними ИС. При этом сторонняя ИС должна реализовывать все те же процедуры, которые реализованы в клиентском приложении **Voice2X Client**:

- Управление пользователями и их межсистемную синхронизацию;
- Проведение аутентификации пользователей;
- Обработку данных о конфигурации Voice2X;
- Реализацию бизнес-логики взаимодействия с службой балансировки нагрузки;
- Захват аудиопотока и приведение его в соответствие с требованиями ASR;
- Информационный обмен со службой бизнес-логики распознавания по протоколу WebSocket;
- Обработку промежуточных и финальных результатов распознавания и заполнение ими экранной формы.

Назначение документа

Документ предназначен для разработчиков информационных систем.

Используемые протоколы

Для обмена сообщениями между вашим приложением и системой Voice2X используются:

- полнодуплексный протокол WebSocket;
- протокол HTTP REST.

Разные типы сообщений служат для выполнения различных действий.

Описание API

В некоторых случаях может быть невозможно проведение интеграции на уровне передачи задач на заполнение протоколов через службу управления задачами (Task Service), описанной в документе Voice2X API. Тогда ИС должна выполнять все функции клиентского приложения **Voice2X Client** и использовать нативную интеграцию.

При нативной интеграции вашему приложению потребуется взаимодействовать со службой аутентификации, службой балансировки нагрузки, самостоятельно обрабатывать звук, отправлять его рекомендованному сервису бизнес-логики распознавания и интерпретировать его ответы. Для упрощения разработки и поддержки вашего решения мы настоятельно рекомендуем использовать интеграцию через службу управления задачами.

Voice2X использует TSL шифрование трафика при подключении по протоколам WebSockets Secure (WSS) и HTTPS.

Быстрый старт

Пример иллюстрирует работу с единственной службой бизнес-логики распознавания.
Функциональность примера можно проверить с помощью плагина Simple WebSocket Client или приложения Postman.

Запрос конфигурации

Обратитесь к сервису Database Service с discovery-запросом (rest) для получения конфигурации системы.

Получение Access Token

Обратитесь к сервису Authentication Service (rest) с предъявлением учётных данных пользователя (логин и SHA256 от пароля). Сохраните Access Token для последующего предъявления в запросах.

Access Token имеет ограниченный срок действия. По истечению этого срока вы можете повторно запросить Access Token или перевыпустить его с предъявлением Refresh Token.

Подключение

Подключите ваше приложение к известному вам сервису бизнес-логики распознавания (WebSocket).

Авторизация

Обмен сообщениями между вашим приложением и сервисом бизнес-логики распознавания требует аутентификации (WebSocket).

Перед выполнением запросов ИС должна авторизоваться, послав сообщение с AccessToken пользователя системы, от лица которого ведётся заполнение протокола.

```
{
   "MessageType": 28,
   "Data":{
      "AccessToken":"••••••"
   }
}
```

Сбор данных сервером

Перед запуском распознавания (WebSocket) также должно быть в явном виде указано, должен ли сервер сохранять звук, переданный пользователем.

```
{
    "MessageType": 13,
    "Data": {
        "SendingMode": false
    }
}
```

Статистические данные

Также перед запуском распознавания должны быть указаны данные о клиентском ПО и пользователе, который производит диктовку (WebSocket).

```
{
    "MessageType": 29,
    "Data" :
    {
        "TimeZone" : "+03:00",
        "AppName" : "My New App",
        "Version" : "1.0.001",
        "RegionCode" : "78",
        "OrganisationFullName" : "ООО «Название организации, в котором работает пользователь»",
        "OrganisationOid" : "123456789012",      // идентификатор организации
        "OrganisationPsrn" : "987654321098765", //ОГРН организации
        "Room" : "102",                          //помещение, где установлено ПО
        "OperatingSystem" : "operating system", // ОС или версия браузера, если работа ведётся через веб-
клиент
        "HostName" : "hostname_1",               //Имя устройства, с которого подключается пользователь
        "Account" : "Account_1",                 //Учётная запись в ОС. Передавать, если не пусто.
        "AccountUserName" : "Иванов И.И.",       //ФИО учётной записи в ОС. Передавать, если не пусто.
        "UserId" : "ac3d8acf-abe7-4a60-a864-bf4e51bdaed3", // идентификатор пользователя
        "UserName" : "Иван Иванович Иванов",     //ФИО пользователя
        "UserIian" : "123-789-654", //СНИЛС пользователя
        "UserSpecialisation" : "Рентгенолог"     //специализация пользователя
    }
}
```

Подготовка описания формы

При использовании Voice2X для голосового заполнения формы, вами должно быть подготовлено описание формы в строго заданном формате.

Описание, в частности, включает в себя название формы, её уникальный идентификатор, перечень полей и их атрибутов. При заполнении формы порядок следования полей и их группировка будут соответствовать тому порядку, который приведён в описании формы.

Запуск распознавания

В простейшем случае запуск распознавания запускается сразу после получения описания формы (WebSocket).

Передача описания формы ведётся в сообщении с `MessageType = 106`.

```
{
    "MessageType": 106,
    "Data": {
        "Key": "a0bc1744-010c-4414-84e6-acabeff7b517",
        "Name": "Форма №1",
        "Childs": [
            {
                "DateTimeFormat": "dd.MM.yyyy",
                "Type": 4,
                "Synonyms": [
                    "Дата осмотра",
                    "Дата проведения осмотра"
                ],
                "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
            }
        ],
        "CultureCode": "ru-RU"
    }
}
```

В качестве подтверждения получения запроса и начала распознавания **Dictation Service** возвращает сообщение с `MessageType = 2`.

```
{
    "MessageType": 2,
    "Data": {
        "Mode": 3,
        "State": 1,
        "Info": "a0bc1744-010c-4414-84e6-acabeff7b517"
    }
}
```

Диктовка и обработка результата

После этого должен быть передан бинарный поток аудиоданных (WebSocket), который соответствует одной из фраз:

- Дата осмотра пятого января;
- Дата проведения осмотра пятое января.

В ответ Voice2X присылает сообщение:

```
{
  "MessageType": 201,
  "Data": {
    "Type": 4,
    "Value": "05.01.2024",
    "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
  }
}
```

Остановка распознавания

Для остановки распознавания формы следует отправить сообщение с `MessageType = 107` (WebSocket).

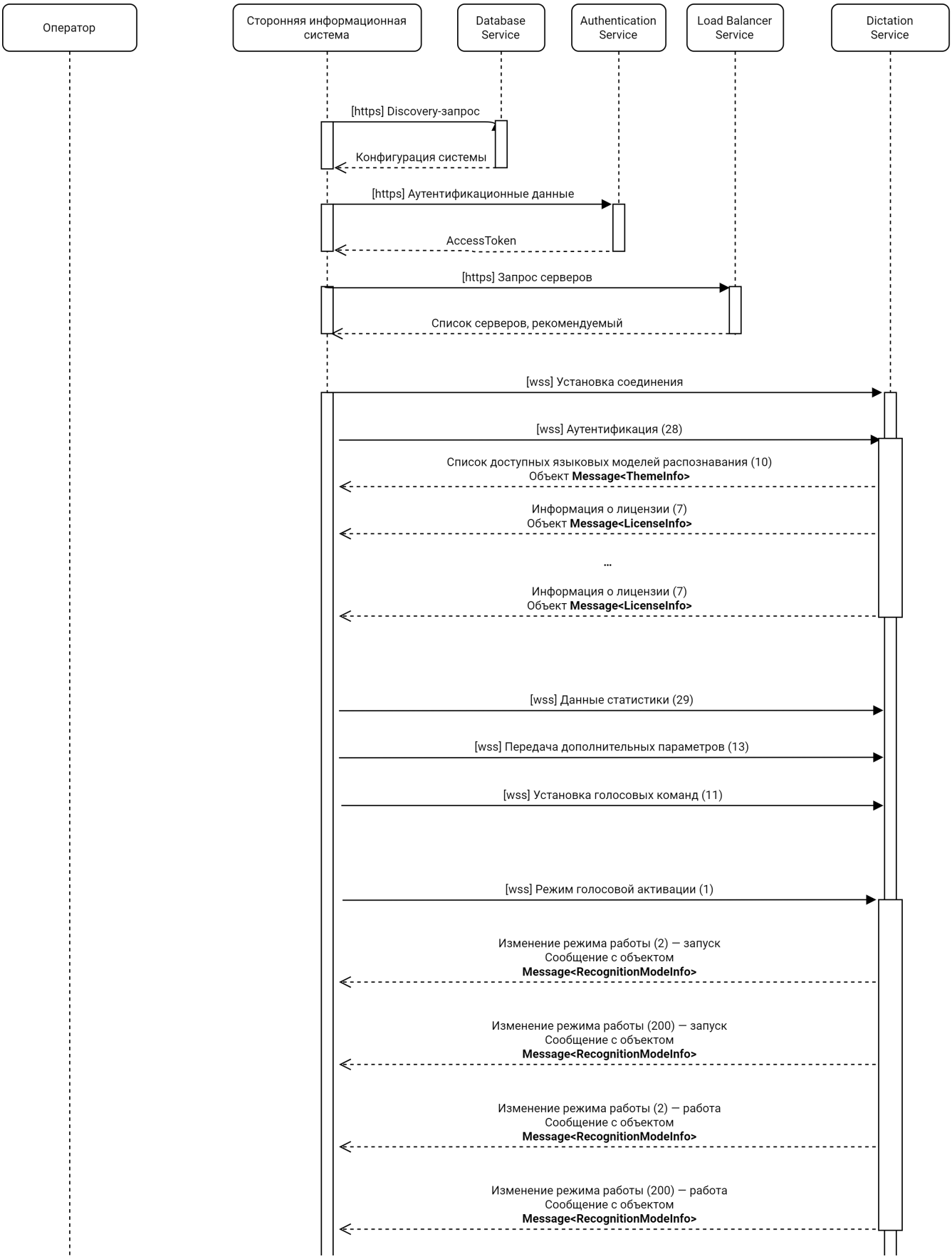
```
{
  "MessageType": 107
}
```

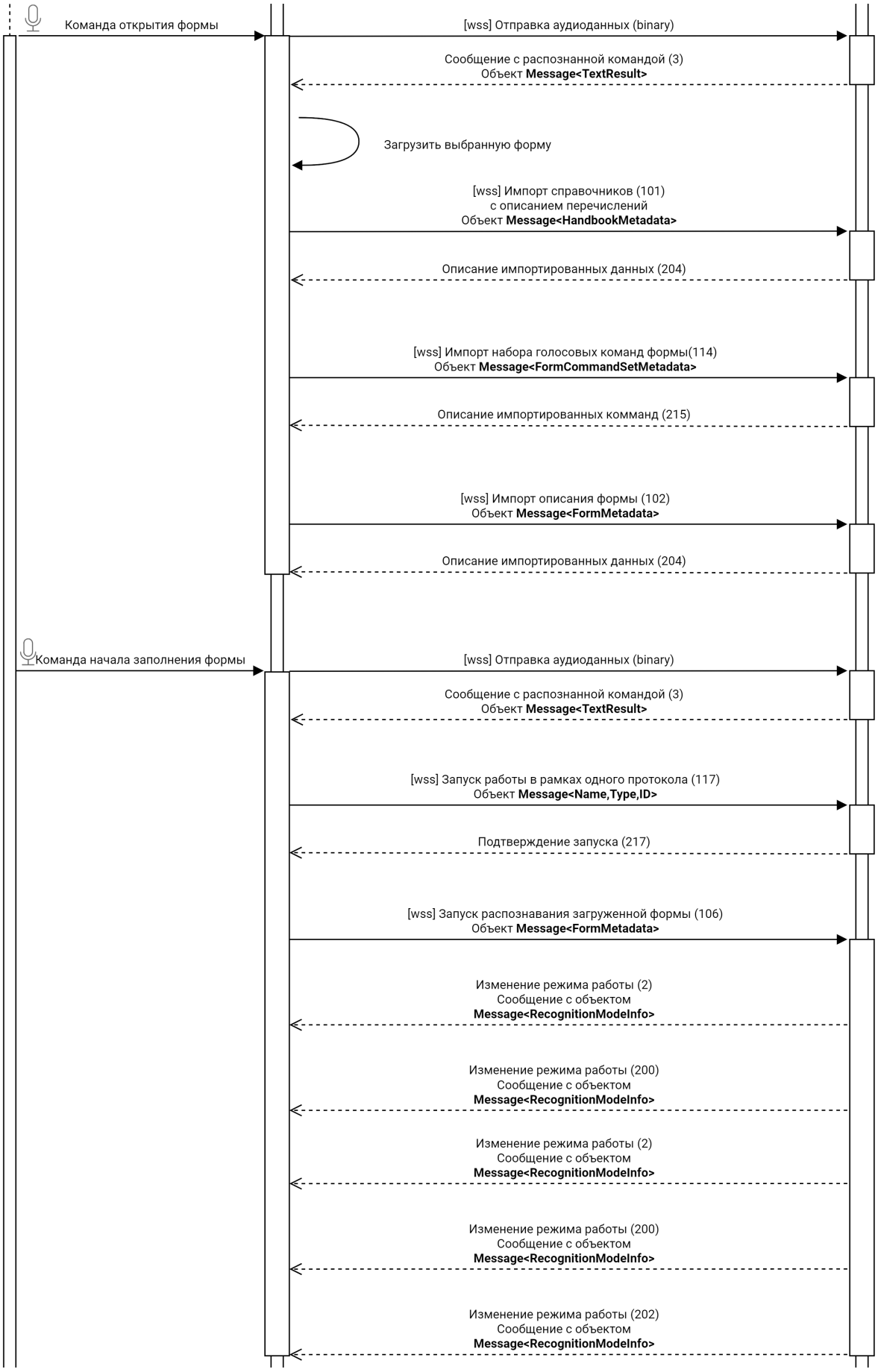
В ответ **Voice2X Client** присылает сообщение:

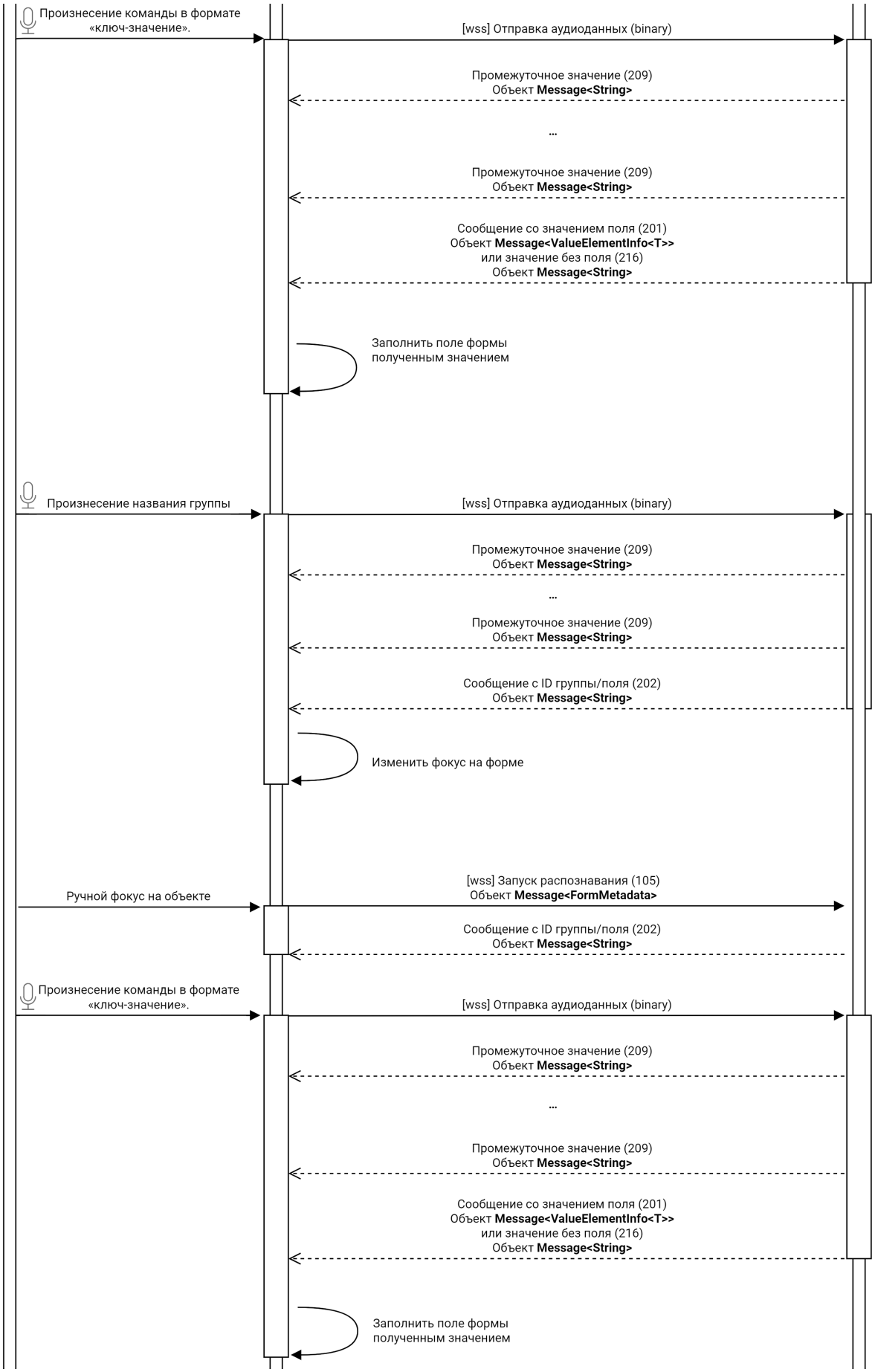
```
{
  "MessageType": 2,
  "Data": {
    "Mode": 0,
    "State": 1
  }
}
```

Диаграмма последовательности

Схема взаимодействия при прямом подключении (рис. 1):




Команда открытия формы
[wss] Отправка аудиоданных (binary)
Сообщение с распознанной командой (3)
Объект **Message<TextResult>**
Загрузить выбранную форму
[wss] Импорт справочников (101)
с описанием перечислений
Объект **Message<HandbookMetadata>**
Описание импортированных данных (204)
[wss] Импорт набора голосовых команд формы(114)
Объект **Message<FormCommandSetMetadata>**
Описание импортированных комманд (215)
[wss] Импорт описания формы (102)
Объект **Message<FormMetadata>**
Описание импортированных данных (204)
Команда начала заполнения формы
[wss] Отправка аудиоданных (binary)
Сообщение с распознанной командой (3)
Объект **Message<TextResult>**
[wss] Запуск работы в рамках одного протокола (117)
Объект **Message<Name,Type,ID>**
Подтверждение запуска (217)
[wss] Запуск распознавания загруженной формы (106)
Объект **Message<FormMetadata>**
Изменение режима работы (2)
Сообщение с объектом
Message<RecognitionModeInfo>
Изменение режима работы (200)
Сообщение с объектом
Message<RecognitionModeInfo>
Изменение режима работы (2)
Сообщение с объектом
Message<RecognitionModeInfo>
Изменение режима работы (200)
Сообщение с объектом
Message<RecognitionModeInfo>
Изменение режима работы (202)
Сообщение с объектом
Message<RecognitionModeInfo>


Произнесение команды в формате «ключ-значение».
[wss] Отправка аудиоданных (binary)
Промежуточное значение (209)
Объект **Message<String>**
...
Промежуточное значение (209)
Объект **Message<String>**
Сообщение со значением поля (201)
Объект **Message<ValueElementInfo<T>>**
или значение без поля (216)
Объект **Message<String>**
Заполнить поле формы полученным значением
Произнесение названия группы
[wss] Отправка аудиоданных (binary)
Промежуточное значение (209)
Объект **Message<String>**
...
Промежуточное значение (209)
Объект **Message<String>**
Сообщение с ID группы/поля (202)
Объект **Message<String>**
Изменить фокус на форме
Ручной фокус на объекте
[wss] Запуск распознавания (105)
Объект **Message<FormMetadata>**
Сообщение с ID группы/поля (202)
Объект **Message<String>**
Произнесение команды в формате «ключ-значение».
[wss] Отправка аудиоданных (binary)
Промежуточное значение (209)
Объект **Message<String>**
...
Промежуточное значение (209)
Объект **Message<String>**
Сообщение со значением поля (201)
Объект **Message<ValueElementInfo<T>>**
или значение без поля (216)
Объект **Message<String>**
Заполнить поле формы полученным значением

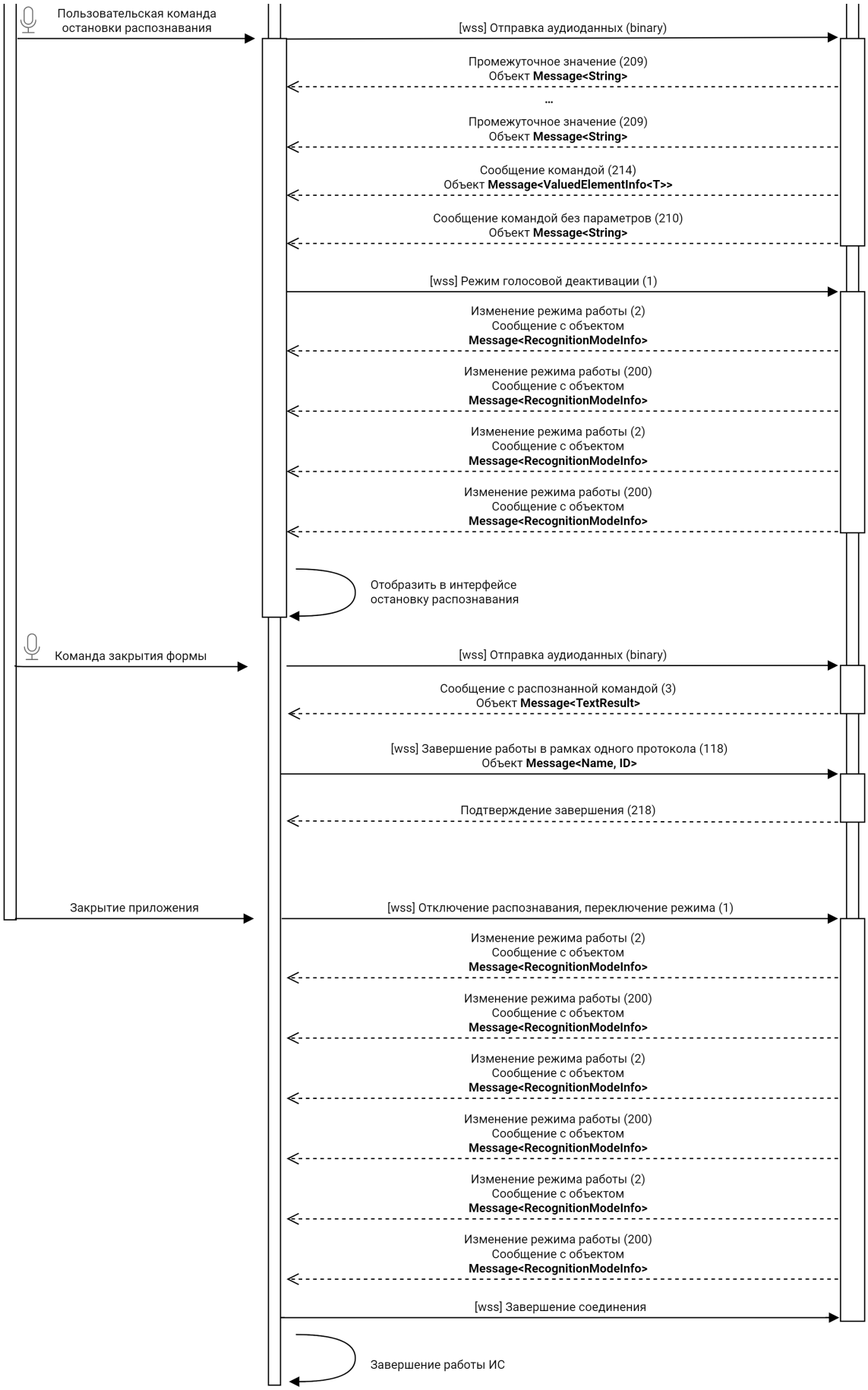


Рисунок 1 – Диаграмма последовательности при предварительной отправке формы

Таким образом, в общем случае от ИС требуется

- получить конфигурацию Voice2X,
- получить Access Token,
- получить рекомендованный сервис бизнес-логики распознавания,
- передать параметры распознавания (сбора данных),
- изменить режим работы,
- прислать справочники и голосовые команды,
- прислать описание формы,
- запустить распознавание по команде пользователя,
- прислать аудиоданные в бинарных посылках,
- обработать приходящие от Voice2X сообщения,
- прислать сообщение о смене фокуса, если пользователь вручную переключился в поле,
- остановить распознавание после того, как все поля заполнены.

Получение конфигурации системы

Voice2X состоит из набора служб, каждая из которых выполняет определённую задачу. При нативной интеграции ИС может быть важно получить сетевую конфигурацию всех служб, содержащихся в конфигурации.

Набор служб и их URI Voice2X зависят от конфигурации и может различаться. Конфигурация может содержать неполный набор служб.

Для получения конфигурации системы вам требуется знать расположение сервиса взаимодействия с реляционной БД (DBService). Далее обозначим URI этой службы как {DBServiceAddresses}. Для выполнения запроса не требуется аутентификация.

GET {DBServiceAddresses}/api/v1/Settings/ConnectionAddresses

Назначение

Получение конфигурации системы.

Заголовок запроса

Авторизация не требуется.

Поля запроса

Нет.

Пример запроса:

```
curl --location 'https://srvdbs.speechpro.com:39444/api/v1/Settings/ConnectionAddresses'
```

Поля ответа

В структуре `Data` возвращается набор URI для каждого компонента Voice2X:

- `AuthenticationServiceAddresses` — служба аутентификации;
- `BalancerAddresses` — служба балансировки нагрузки;
- `DBServiceAddresses` — служба взаимодействия с реляционной БД;
- `EDBServiceAddresses` — служба взаимодействия с нереляционной БД;
- `FeedbackServiceAddresses` — служба отправки обратной связи;
- `ReportServiceAddresses` — служба формирования отчётов;
- `StatisticServiceAddresses` — служба статистики;
- `TaskServiceAddresses` — служба управления задачами.

Пример:


```
{
    "Data": {
        "BalancerAddresses": [
            "https://srvlbs.speechpro.com:39255"
        ],
        "FeedbackServiceAddresses": [
            "https://srvfbs.speechpro.com:5000"
        ],
        "StatisticServiceAddresses": [
            "https://srvss.speechpro.com:64995"
        ],
```

```
        "AuthenticationServiceAddresses": [
            "https://srvauth.speechpro.com:39300"
        ],
        "DBServiceAddresses": [
            "https://srvdbs.speechpro.com:39444"
        ],
        "EDBServiceAddresses": [
            "https://srvedbs.speechpro.com:39600"
        ],
        "TaskServiceAddresses": [
            "https://srvts.speechpro.com:39500"
        ],
        "ReportServiceAddresses": [
            "https://srvrs.speechpro.com:39650"
        ]
    },
    "Error": null
}
```


Обратите внимание, что службы могут иметь несколько адресов. Вы должны корректно обрабатывать их, перебирая по очереди при недоступности.

Авторизация и аутентификация

Обмен сообщениями между вашим приложением и Voice2X через https и websoket требует авторизации и аутентификации.

Для аутентификации требуется предъявить логин пользователя и хэш пароля, преобразованный по алгоритму SHA256, и переведённый в верхний регистр. Пользователь должен быть однозначно идентифицируемым. Обозначим URI службы аутентификации как {DBServiceAddresses}, полученный в запросе конфигурации системы.

POST {AuthenticationServiceAddresses}/api/v1/Account/Login

Назначение

Аутентификация пользователя и получение токенов AccessToken и RefreshToken.

Заголовок запроса

Content-Type: application/json. Авторизация не требуется.

Тело запроса

Тело запроса должно содержать json-структуру с данными пользователя.

Поле	Тип	Примечание
Login	String	Имя аккаунта, зарегистрированное для использования внешней информационной системой
Password	String	SHA256 от пароля, зарегистрированного для использования внешней информационной системой, приведённое к верхнему регистру

Пример запроса:

```
curl --location 'https://srvauth.speechpro.com:39300/api/v1/Account/Login' \
--header 'Content-Type: application/json' \
--data '{
    "Login":"user",
    "Password":"0CAC6F785283427E764801EF8F2B5845733FF513F59F88765706F9C5B7C14D67"
}'
```

Поля ответа

В структуре `Data` возвращается набор данных:

- `TokenModel` — структура с токенами;
- `UserId` — уникальный идентификатор пользователя;
- `Permissions` — код набора привилегий пользователя в Voice2X.

Структура с токенами содержит:

- `AccessToken`;
- `RefreshToken`;
- `AccessTokenValidTo` — срок действия токена `AccessToken`. По истечении этого времени использование токена `AccessToken` будет невозможно. Потребуется повторная авторизация или перевыпуск токена без предъявления логина и пароля;
- `RefreshTokenValidTo` — срок действия токена `RefreshToken`. По истечении этого времени использование токена `RefreshToken` будет невозможно. Потребуется повторная авторизация.

Пример:

```
{
    "Data": {
        "TokenModel": {
            "AccessToken": "••••••",
```

```
            "RefreshToken": "••••••",
            "AccessTokenValidTo": "2024-07-02T07:24:37.1680643Z",
            "RefreshTokenValidTo": "2024-07-02T19:00:00Z"
        },
        "UserId": "4e753920-4bf0-4630-9371-79531b1eaf76",
        "Permissions": 18446744073709551615
    }
}
```

Выбор службы бизнес-логики распознавания

Описание принципов работы со службой балансировки нагрузки

Voice2X может содержать несколько служб бизнес-логики распознавания речи. В таком случае перед подключением к одной из служб требуется запросить у службы балансировки нагрузки рекомендуемую.

Каждая служба бизнес-логики распознавания речи периодически отправляет службе балансировки нагрузки данные о себе и о нагрузке, которую создают подключённые к ней клиенты. Служба балансировки нагрузки ведёт учёт всех служб бизнес-логики распознавания речи и выявляет менее загруженную для подключения в данный момент (рис. 2).

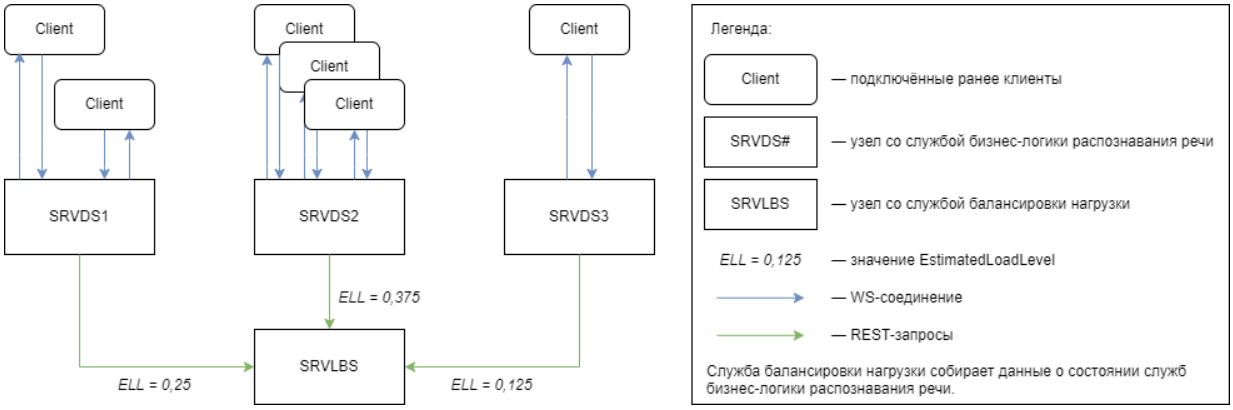


Рисунок 2 — Сбор службой балансировки нагрузки данных о состоянии служб бизнес-логики распознавания речи

В ответ на запрос служба балансировки нагрузки возвращает адрес службы бизнес-логики распознавания речи, к которой рекомендует подключаться в данный момент (менее загруженную) и общий список служб со значением моментальной нагрузки (рис. 3).

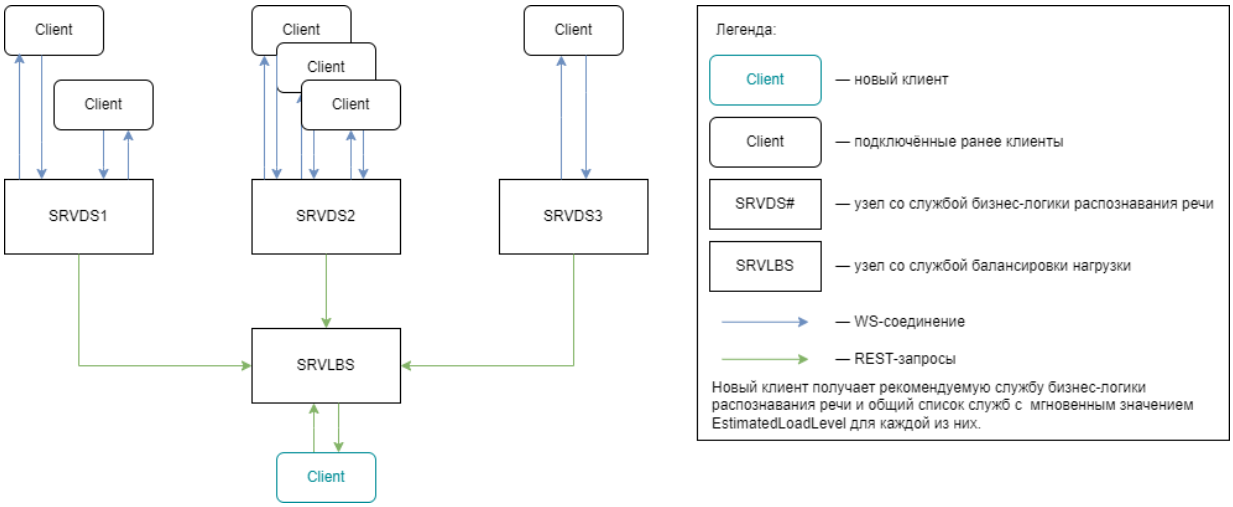


Рисунок 3 — Запрос клиентом рекомендованной службы для подключения

Клиент подключается к рекомендованной службе бизнес-логики распознавания речи (рис. 4).

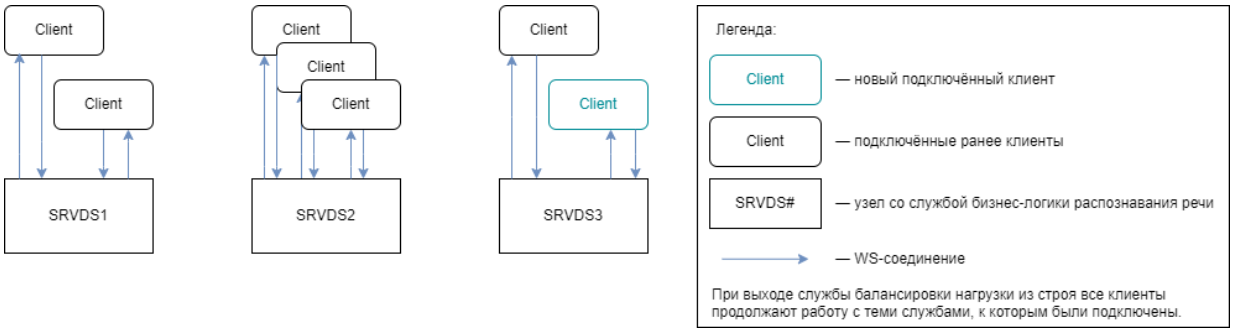


Рисунок 4 — Подключение клиента к рекомендованной службе

Так как нагрузка на службы распознавания речи меняется со временем (диктующие клиенты отключаются, подключаются новые клиенты), то клиентское приложение должно обновлять данные о службах, которые рекомендуется использовать для подключения (рис. 5).

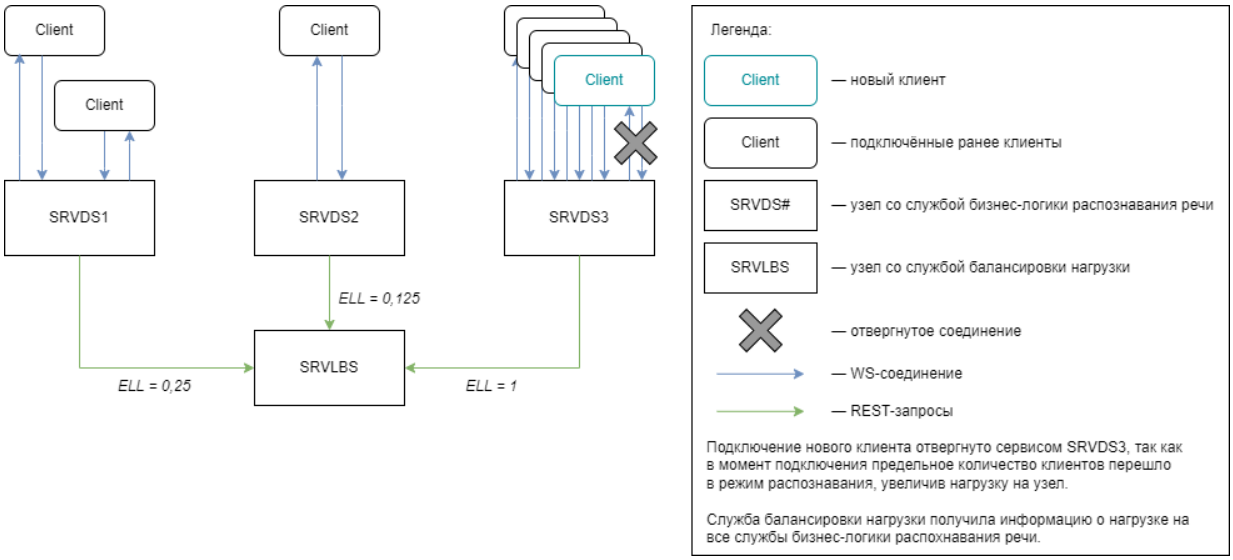


Рисунок 5 — Отключение клиента

В приложении **Voice2X Client** реализуется следующая логика работы:

- При первом подключении к службе бизнес-логики распознавания речи приложение получает и сохраняет ответ службы балансировки нагрузки (рис. 3).
- Если служба бизнес-логики распознавания речи разорвала соединение, а сохранённый ответ службы балансировки нагрузки не устарел (время жизни регулируется настройками, по умолчанию равно 10 минутам), то клиентское приложение пробует подключиться к следующей наименее загруженной службе в порядке возрастания `EstimatedLoadLevel` (рис. 6). Служба может отвергать подключение клиента, если на момент подключения максимальное возможное количество подключенных клиентов перешло в режим распознавания из режима отключенного распознавания или если большое количество клиентов подключилось к службе, используя сохранённые ответы службы балансировки нагрузки, полученные ранее.

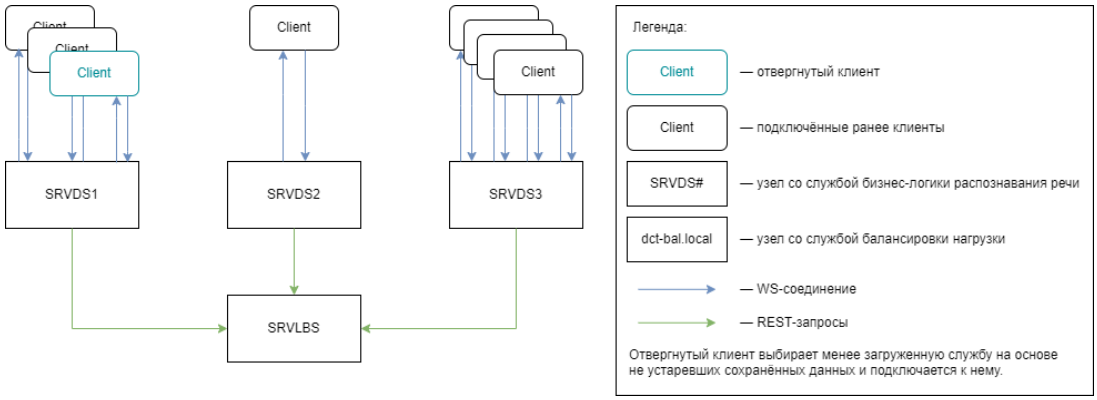


Рисунок 6 – Подключение клиента к свободной службе без получения рекомендованной

- Если ответ от службы балансировки нагрузки устарел, то клиентское приложение снова запрашивает у неё список служб бизнес-логики распознавания речи, и подключается к одной из них (рис. 7).

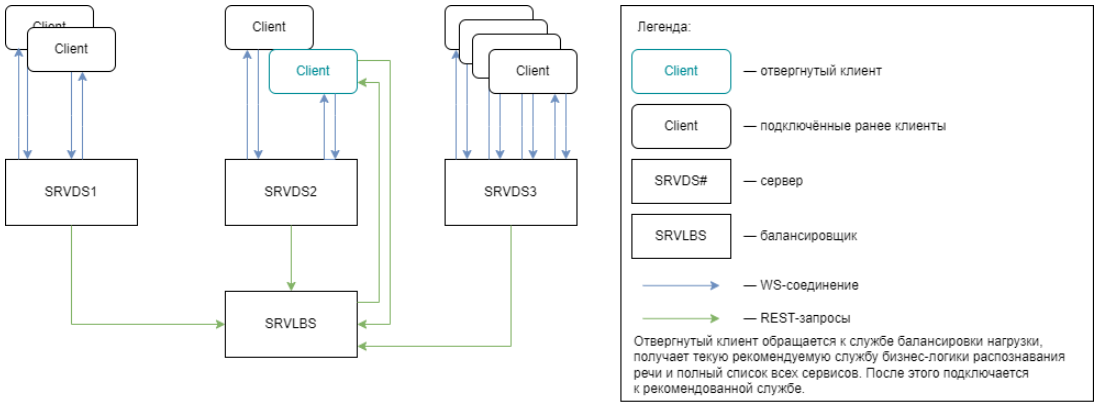


Рисунок 7 – Подключение клиента к свободной службе бизнес-логики распознавания речи после получения у службы балансировки нагрузки рекомендованной службы

Важно, что список служб бизнес-логики распознавания речи позволяет судить о загруженности служб, входящих в систему, в текущий момент времени. Поэтому он должен регулярно обновляться.

Служба балансировки нагрузки не проксирует через себя никакой трафик.
После получения адреса рекомендованной службы и полного списка служб клиент работает с ними напрямую.

Такой подход позволяет исключить влияние надёжности службы балансировки нагрузки на систему:

1. При выходе службы балансировки нагрузки из строя в тот момент, когда клиент подключён к службе бизнес-логики распознавания речи и работает, клиент продолжит работу в штатном режиме, так как работа ведётся напрямую (рис. 8).

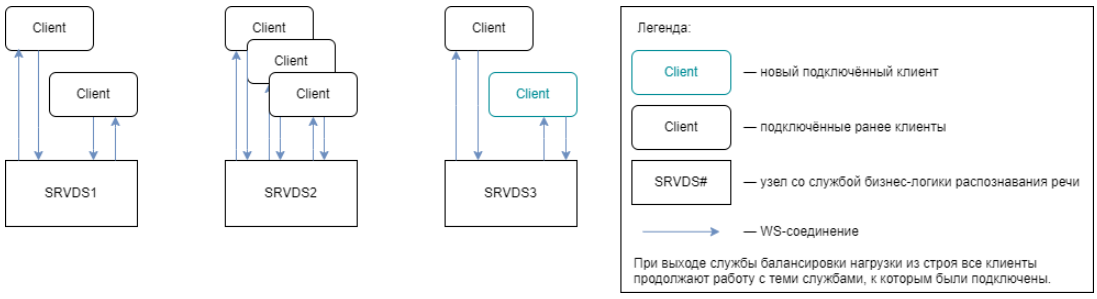


Рисунок 8 – Выход из строя службы балансировки нагрузки

2. При выходе из строя службы балансировки нагрузки и при перегрузке службы бизнес-логики распознавания речи запросами клиент будет пробовать подключиться к следующей известной ему службе бизнес-логики распознавания речи (рис. 6).

Получение адреса службы бизнес-логики распознавания

Обозначим URI службы балансировки нагрузки как {BalancerAddresses}, полученный в запросе конфигурации системы.

POST {BalancerAddresses}/api/v1/servers?clientId=

Назначение

Получение сведений о сервисах бизнес-логики распознавания речи, подключённых к службе балансировки нагрузки, и их свойствах.

Параметры запроса

Параметр	Тип	Примечание
clientId	`String`	Уникальный идентификатор пользователя, который должен совпадать с UserId, полученным при аутентификации. Если параметр не передан, то ответ не содержит рекомендованный для подключения сервер (блок `RecommendedServer`), а содержит лишь полный список серверов (блок `ServerList`).

Пример:

```
curl --location --request POST 'https://srvlbs.speechpro.com:39255/api/v1/servers/?clientId=4e753920-4bf0-4630-9371-79531b1eaf76 \
--header 'Authorization: ••••••'
```

Поля ответа

Поле	Тип	Примечание
TimeStamp	`DateTime`	Текущее время на балансировщике нагрузки
RecommendedServer	`ServerInfo`	Рекомендуемый к использованию сервер
ServerList	Список `ServerInfo`	Список серверов, подключённых к балансировщику нагрузки

Класс ServerInfo имеет следующие поля:

Поле	Тип	Примечание
ServerId	`String`	Уникальный идентификатор сервера
ConnectionString	`String`	Строка для подключения к серверу формата `<адрес сервера>;<WebSocket порт>;<Rest порт>`
CanAcceptClient	`Boolean`	Может ли сервер принять ещё одного клиента
EstimatedLoadLevel	`Decimal`	Предполагаемая загруженность сервера с учётом клиентов, недавно запросивших список серверов
DeclaredLoadLevel	`Decimal`	Заявленная загруженность сервера активными клиентами
LastUpdateDate	`DateTime`	Время последнего обновления данных о сервере на балансировщике нагрузки
LoadPerClient	`Decimal`	Загруженность сервера одним клиентом

Пример:


```
{
    "Data": {
        "TimeStamp": "2024-07-02T09:05:29.3020837Z",
        "RecommendedServer": {
            "ServerId": "d0ace952-234a-4df5-b6fb-5cabcfe2f8fd",
            "ConnectionString": "srvds1.speechpro.com;34000;39255",
            "CanAcceptClient": true,
            "EstimatedLoadLevel": 0.0,
            "DeclaredLoadLevel": 0.0,
            "LastUpdateDate": "2024-07-02T09:04:46.0349541Z",
            "LoadPerClient": 0.0625
        },
        "ServerList": [
```

```
            {
                "ServerId": "d0ace952-234a-4df5-b6fb-5cabcfe2f8fd",
                "ConnectionString": "srvds1.speechpro.com;34000;39255",
                "CanAcceptClient": true,
                "EstimatedLoadLevel": 0.0,
                "DeclaredLoadLevel": 0.0,
                "LastUpdateDate": "2024-07-02T09:04:46.0349541Z",
                "LoadPerClient": 0.0625
            },
            {
                "ServerId": "5af5b482-87fa-4ea4-acaf-330c2b2fd144",
                "ConnectionString": "srvds2.speechpro.com;34000;39255",
                "CanAcceptClient": true,
                "EstimatedLoadLevel": 0.1,
                "DeclaredLoadLevel": 0.0,
                "LastUpdateDate": "2024-07-02T09:04:00.0494181Z",
                "LoadPerClient": 0.1
            }
        ]
    }
}
```

Подключение к Dictation Service

Подключение внешней ИС должно производиться к службе бизнес-логики распознавания речи, рекомендованной службой балансировки нагрузки с параметрами, содержащимися в поле `ConnectionString`. Если конфигурация системы содержит лишь одну службу бизнес-логики распознавания речи, то служба балансировки нагрузки может не использоваться; тогда подключение должно производиться напрямую к службе бизнес-логики распознавания речи.

По умолчанию у Dictation Service настроены следующие порты:

- для WebSocket (основное подключение): **server_name:34000**;
- для REST (для вспомогательных запросов): **server_name:39255**.

Обратите внимание, что администратор может изменить значения портов. Корректные значения получите в ответ от службы балансировки нагрузки.

В любом случае ИС должна обеспечить:

- захват звука
- передачу звука на сервер
- другие операции (например: работа с балансировщиком, логика обработки автозамен)

После подключения к службе бизнес-логики распознавания речи обязательно передать сообщения

Режимы работы Voice2X

Voice2X поддерживает следующие режимы работы:

0. **Система отключена**.
   Режим работы, в котором распознавание не производится и система бездействует.
   При работе ИС напрямую с **Dictation Service** переключение всех режимов должна выполнять сама ИС.
   Об отключении распознавания **Dictation Service** информирует ИС сообщениями об изменении режима работы и изменении состояния распознавания.

1. **Режим голосовой активации**.
   Режим предназначен для того, чтобы предоставить ИС возможность включить распознавание при помощи голосовой команды и реагировать на голосовые команды, определяемые ею (не путать с пользовательскими командами при заполнении формы).
   В этом режиме результат распознавания приходит только при произнесении пользователем фразы активации распознавания и голосовых команд, которые задаёт ИС. Прочая речь пользователя будет проигнорирована.
   При работе ИС напрямую с **Dictation Service** переключение в этот режим должна выполнять сама ИС.
   О переходе в режим голосовой активации **Dictation Service** информирует ИС сообщениями об изменении режима работы и изменении состояния распознавания.
   В задачу ИС входит реагирование на сообщения при получении сообщения от **Dictation Service** (смена режима работы и выполнения действий, за которыми закреплена зарегистрированная голосовая команда).

2. **Режим слитного распознавания**.
   Режим предназначен для голосового заполнения простого текстового документа (например, заметки в блокноте) и распознавания голосовых команд, заданных ИС.
   ИС должна корректно обрабатывать сообщения с результатом распознавания (текстом, голосовыми командами ИС и командой деактивации распознавания).

3. **Режим заполнения формы**.
   Режим предназначен для голосового заполнения формы по подготовленному вашим приложением описанию формы.
   Распознав в словах пользователя наименование поля и его значение, система отправит соответствующее сообщение в ИС.
   ИС должна корректно обработать передаваемые сообщения со значением заполняемого поля и голосовые команды, определённые при инициализации формы.
   В зависимости от свойств формы может производиться синтез названия полей и распознанных значений, автоматический перевод фокуса и пр.

4. **Резерв**

5. **Офлайн распознавание**.
   Экспериментальная функция. Режим предназначен для распознавания аудиозаписи с разделением дикторов. В поставку не включается и в документации описан не полностью. Для уточнения обратитесь к разработчику.

При получении описания формы и запуске распознавания производится автоматическое переключение режима работы.

Подготовка описания формы и её свойств

При работе в режиме заполнения формы (3) вашему приложению потребуется подготовить и передать в Voice2X описание формы до запуска распознавания речи или одновременно с ним.

Правила описания формы приводятся в разделе Формат описания формы данного.

В зависимости от параметров формы `FormSettings`, передаваемых вместе с описанием, Voice2X может:

- Озвучивать названия полей (включить TTS для озвучивания полей);
- Озвучивать значения полей (включить TTS для озвучивания значений);
- Автоматически переводить фокус на следующее поле при успешном заполнении текущего поля (переходить на следующее поле после заполнения этого поля);
- Останавливать заполнение формы (и распознавание речи) после заполнения последнего поля (останавливать распознавание в конце формы).

Подробная информация о параметрах работы с формой приведена в разделе Класс FormSettings.

Отдельно от описания формы могут быть импортированы общие справочники и команды.

Справочники следует импортировать во все службы бизнес-логики распознавания речи, которые входят в программный комплекс.

Список WS-методов (сообщений)

Любое сообщение, которыми обмениваются Voice2X и информационная система посредством WebSocket, оборачивается в типизированный класс-контейнер `DictationApiMessage<T>`.

Класс-контейнер, в который оборачиваются все сообщения			
Поле	Тип поля	Обязательное	Описание
MessageType	`Int32`	Да	Тип сообщения
Data	зависит от сообщения	Нет	Данные, соответствующие типу сообщения

Сообщения от ИС

После того, как установлено подключение между ИС и **Dictation Service**, производится обмен сообщениями. Сообщения, отправляемые ИС, представляют собой команды и в большинстве случаев подразумевают ответное сообщение.

Сообщения с кодом более 100 используются при работе в режимах заполнения формы (3 и 4). Сообщения с кодом менее 100 используются при работе в режимах голосовой активации, слитного распознавания речи и офлайн-распознавания (1, 2 и 5).

Полный список команд с указанием кода сообщения приведён в таблице 1.

Таблица 1: Список команд

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
1	Изменить режим распознавания	Сообщение 2 и сообщение 200	Не предусмотрено
11	Обновление пользовательских команд	Не предусмотрено	Сообщение 5 и сообщение 205
13	Запрос на сохранение пользовательских данных	Не предусмотрено	Сообщение 5 и сообщение 205
21	Запрос параметров синтезатора речи	Сообщение 22 и/или сообщение 211	Не предусмотрено
23	Изменение настроек синтеза речи	Сообщение 22 и/или сообщение 211	Не предусмотрено
28	Аутентификация сессии	Не предусмотрено	Сообщение 5 и сообщение 205
29	Информация о пользователе и приложении	Не предусмотрено	Сообщение 5 и сообщение 205
101	Предварительный импорт справочников	Сообщение 204	Сообщение 205
102	Предварительный импорт формы	Сообщение 204	Сообщение 205
103	Запрос списка импортированных данных по типу	Сообщение 203	Сообщение 205
104	Запрос описания импортированных данных по ключу	Сообщение 204	Сообщение 205
105	Установка текущего активного элемента или группы	Сообщение 202	Сообщение 205
106	Запуск заполнения формы	Сообщение 200	Сообщение 205
107	Завершение заполнения формы	Сообщение 200	Сообщение 205
110	Изменение параметров заполнения формы	Не предусмотрено	Не предусмотрено
111	Состояние синтезатора речи	Сообщение 211	Сообщение 205
112	Зарезервировано	-	-
113	Предварительный импорт набора голосовых команд	Сообщение 204	Сообщение 205
114	Изменение текущего набора голосовых команд	Сообщение 215	Сообщение 205
115	Зарезервировано	-	-
116	Зарезервировано	-	-

119	Отключение синтеза речи	Не предусмотрено	Не предусмотрено
120	Включение синтеза речи	Не предусмотрено	Не предусмотрено
130	Распознавание в рамках одного документа	Не предусмотрено	Не предусмотрено

По порядку выполнения сообщения можно объединить в группы.

Должны выполняться после установления соединения:

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
13	Запрос на сохранение пользовательских данных	Не предусмотрено	Сообщение 5 и сообщение 205
28	Аутентификация сессии	Не предусмотрено	Сообщение 5 и сообщение 205
29	Информация о пользователе и приложении	Не предусмотрено	Сообщение 5 и сообщение 205

Должны выполняться при голосовой активации распознавания:

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
11	Обновление пользовательских команд	Не предусмотрено	Не предусмотрено
1	Изменить режим распознавания	Сообщение 2 и сообщение 200	Не предусмотрено

Должны выполняться перед запуском распознавания формы:

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
101	Предварительный импорт справочников	Сообщение 204	Сообщение 205
102	Предварительный импорт формы	Сообщение 204	Сообщение 205
113	Предварительный импорт набора голосовых команд	Сообщение 204	Сообщение 205
130	Распознавание в рамках одного документа	Не предусмотрено	Не предусмотрено

Для запуска распознавания формы следует выполнять

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
106	Запуск заполнения формы	Сообщение 200	Сообщение 205

Должны выполняться во время распознавания формы

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
105	Установка текущего активного элемента или группы	Сообщение 202	Сообщение 205
110	Изменение параметров заполнения формы	Не предусмотрено	Не предусмотрено
114	Изменение текущего набора голосовых команд	Сообщение 215	Сообщение 205

Для остановки распознавания формы следует выполнять

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
107	Завершение заполнения формы	Сообщение 200	Сообщение 205

В любой момент в режимах 0, 1, 3 и 4 можно выполнять

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
103	Запрос списка импортированных данных по типу (если был импорт)	Сообщение 203	Сообщение 205
104	Запрос описания импортированных данных по ключу (если был импорт)	Сообщение 204	Сообщение 205
105	Установка текущего активного элемента или группы (если форма загружена)	Сообщение 202	Сообщение 205
110	Изменение параметров заполнения формы (если был импорт)	Не предусмотрено	Не предусмотрено
111	Состояние синтезатора речи	Сообщение 211	Сообщение 205
119	Отключение синтеза речи	Не предусмотрено	Не предусмотрено
120	Включение синтеза речи	Не предусмотрено	Не предусмотрено

В любой момент в режимах 0, 1 и 2 можно выполнять

Код сообщения	Наименование команды	Подтверждение	Сообщение при ошибке
1	Изменить режим распознавания	Сообщение 2 и сообщение 200	Не предусмотрено
13	Запрос на сохранение пользовательских данных	Не предусмотрено	Сообщение 5 и сообщение 205
21	Запрос параметров синтезатора речи	Сообщение 22 и/или сообщение 211	Не предусмотрено
23	Изменение настроек синтеза речи	Сообщение 22 и/или сообщение 211	Не предусмотрено

MessageType 1. Изменить режим распознавания

ИС должна управлять режимом распознавания, так Voice2X как может работать не только в режимах заполнения формы (3 и 4), но и в других режимах.

Переключение из режима голосовой активации в режим распознавания (слитного или формы) и назад так же входит в круг задач ИС. Voice2X лишь оповестит о том, что была произнесена фраза голосовой активации или остановки распознавания.

При помощи данного сообщения ИС переключает текущий режим распознавания.

Запрос (секция `Data`) должен содержать поле `Type`:

Запрос на сохранение пользовательских данных			
Поле	**Тип поля**	**Обязательное**	**Описание**
Type	Int32	Да	Режим распознавания: **0** – распознавание отключено; **1** – голосовая активация; **2** – слитное распознавание речи; **3** – свободное заполнение формы; **4** – последовательное заполнение формы; **5** – офлайн распознавание (распознавание записей)

```
{
   "MessageType": 1,
   "Data":{
      "Type":"2"
   }
}
```

В ответ Voice2X отправляет серию уведомлений об изменении режима работы и уведомления об изменении режима распознавания: отключение текущего режима, попытка включения выбранного режима и результат переключения.

MessageType 11. Обновление пользовательских команд

При помощи данного сообщения можно изменить команды голосовой активации распознавания и пользовательские команды, не связанные с формой.

> Обратите внимание: данные пользовательские команды работают в режиме слитного распознавания. При заполнении формы (режимы 3 и 4) эти пользовательские команды игнорируются, но обрабатываются пользовательские команды, переданные в описании заполняемой формы, импортированные дополнительно или переданные дополнительно к описанию формы.

В запросе должен быть передан массив голосовых команд, описываемых классом `CommandInfo`:

Класс описания голосовых команд			
Поле	**Тип поля**	**Обязательное**	**Описание**
Id	`String`	Да	Уникальный идентификатор голосовой команды. Он возвращается в ответе
Type	`Int32`	Да	Тип голосовой команды: **0** – зарезервировано (не использовать); **1** – команда голосовой активации распознавания; **2** – команда голосовой деактивации распознавания; **3** – пользовательская команда
Pattern	`String`	Да	Произносимый пользователем текст

```
{
   "MessageType": 11,
   "Data":{
      "Commands":
      [
          {
              "id":"Start",
              "Type":"1",
              "Pattern":"запустить распознавание"
         },
                {
              "id":"Stop",
              "Type":"2",
              "Pattern":"остановить распознавание"
         },
          {
              "id":"Test",
              "Type":"3",
              "Pattern":"моя команда"
          }
      ]
   }
}
```

Ответа на сообщение не предусмотрено.

В режиме `Mode=1` Voice2X присылает результат распознавания только при произнесении фразы, указанной как `Type=1`.

После этого ИС должна перевести Voice2X в требуемый для работы режим (2, 3 или 4).

В режиме `Mode=2` Voice2X в качестве результата распознавания может прислать распознанную пользовательскую команду или команду деактивации распознавания. В случае получения команды деактивации ИС должна вернуть Voice2X в исходный режим.

MessageType 13. Запрос сохранения пользовательских данных

Voice2X может сохранять звук и распознанный текст. В дальнейшем эти данные могут использоваться для улучшения распознавания речи. ИС может принудительно отключить сбор и хранение данных.

Запрос (секция `Data`) должен содержать поле `SendingMode`:

Запрос на сохранение пользовательских данных			
Поле	Тип поля	Обязательное	Описание
SendingMode	Boolean	Да	Тип кешированных данных: **true** – разрешено сохранять звук и распознанный текст на сервере; **false** – запрещено сохранять звук и распознанный текст на сервере

```
{
    "MessageType": 13,
    "Data": {
        "SendingMode":true
    }
}
```

Ответа на сообщение не предусмотрено.

MessageType 21. Настройки синтезатора речи

В любой момент времени ИС может запросить у Voice2X настройки, используемые синтезатором речи.

```
{
   "MessageType":21,
   "Data":{

   }
}
```

Voice2X возвращает сообщение со списком параметров синтезаторов речи.

MessageType 23. Изменение настроек синтезатора речи

Информационная система может изменить текущий голос синтезатора речи и\или его скорость.

Название синтезатора должно быть выбрано из списка доступных в Voice2X, а скорость должна принимать значения от 0.5 до 2.

```
{
   "MessageType": 23,
   "Data":{
      "SelectedVoice":{
         "Name":"Александр"
      },
      "Speed":1.4
   }
}
```

Здесь Александр — один из синтезаторов речи, имеющихся на сервере. Указан в качестве примера. На реальном сервере может быть другое название синтезатора речи.

Voice2X возвращает сообщение со списком параметров синтезаторов речи.

MessageType 28. Аутентификация сессии

После установления соединения ИС обязана пройти аутентификацию в Voice2X, предъявив AccessToken пользователя.

Поле `Data` сообщения должно содержать следующие поля:

Запрос на аутентификацию			
Поле	**Тип поля**	**Обязательное**	**Описание**
AccessToken	String	Да	AccessToken, полученный у службы аутентификации

```
{
    "MessageType": 28,
    "Data": {
        "AccessToken": "••••••"
    }
}
```

Если не предъявить токен, то на любое сообщение Voice2X вернёт ошибку «Не передан AccessToken».

MessageType 29. Информация о пользователе и приложении

После установления соединения ИС обязана передать информацию о клиентском приложении, пользователе и, по возможности, аккаунте ОС.

Поле `Data` сообщения должно содержать следующие поля:

Сообщение с информацией о пользователе и ПО			
Поле	**Тип поля**	**Обязательное**	**Описание**
TimeZone	String	Да	Часовой пояс, в котором находится клиентское приложение (+03:00)
AppName	String	Да	Название клиентского приложения
Version	String	Да	Версия клиентского приложения
RegionCode	String	Нет	Код региона РФ, в котором установлено клиентское ПО
OrganisationFullName	String	Да	Полное название компании, в котором установлено клиентское ПО
OrganisationOid	String	Да	Идентификатор компании, в котором установлено клиентское ПО
OrganisationPsrn	String	Нет	ОГРН компании, в котором установлено клиентское ПО
Room	String	Нет	Название или номер кабинета, в котором установлено клиентское ПО
OperatingSystem	String	Да	Название операционной системы или версия браузера
HostName	String	Нет	Имя устройства или имя хоста, на котором установлено клиентское ПО
Account	String	Нет	Название учётной записи в ОС
AccountUserName	String	Нет	ФИО учётной записи в ОС
UserId	String	Да	ID учётной записи пользователя
UserName	String	Да	ФИО пользователя
UserIian	String	Да	СНИЛС пользователя
UserSpecialisation	String	Нет	Специализация пользователя

```
{
    "MessageType": 29,
    "Data" :
    {
        "TimeZone" : "+03:00",
        "AppName" : "My New App",
        "Version" : "1.0.001",
        "RegionCode" : "78",
        "OrganisationFullName" : "ООО «Название организации, в котором работает пользователь»",
        "OrganisationOid" : "123456789012",
        "OrganisationPsrn" : "987654321098765",
        "Room" : "102",
        "OperatingSystem" : "operating system",
        "HostName" : "hostname_1",
        "Account" : "Account_1",
        "AccountUserName" : "Иванов И.И.",
        "UserId" : "ac3d8acf-abe7-4a60-a864-bf4e51bdaed3",
        "UserName" : "Иван Иванович Иванов",
        "UserIian" : "123-456-789 01",
        "UserSpecialisation" : "Рентгенолог"
    }
}
```

MessageType 101. Импорт справочника

Большинство форм содержит, помимо простых полей, заполняемых распознанными значениями, ссылочные поля, связанные с соответствующими справочниками. Для голосового заполнения таких полей следует передавать все значения из связанной таблицы-справочника в формате «ключ-значение». Такой же подход следует применять и к полям-перечислениям (см. в примере поле “Пол”). Импорт справочников следует производить на все серверы распознавания, если их несколько.

Предварительный импорт производится сообщением, содержащим класс `HandbookMetadata`.

Класс описания данных справочника			
Поле	**Тип поля**	**Обязательно**	**Описание**
HandbookKey	`String`	Да	Ключ-индентификатор справочника – уникальное значение, однозначно идентифицирующий этот справочник. Значение должно содержать только символы латинского алфавита, нижнее подчеркивание и цифры. UUID и GUID использовать нельзя из-за символа «-»
Name	`String`	Нет	Наименование
AvailableValues	Список элементов состоящий из `KeyValuePair<string, string>`	Да	Список возможных значений в формате “ключ-значение”

Ниже приведён пример сообщения предварительной инициализации справочника и описание полей передаваемых данных.

```
{
  "MessageType": 101,
  "Data":
    {
      "HandbookKey": "38d61a2b_cde8_4d17_b806_60c5fa8c8769",
      "Name": "Пол",
      "AvailableValues": [
          {
               "Key": "0",
               "Value": "Сухой"
          },
          {
               "Key": "1",
               "Value": "Мокрый"
          }
      ]
    }
}
```

В ответ Voice2X отправляет сообщение с описанием предварительно импортированных данных.

MessageType 102. Импорт формы

Описание формы можно импортировать до запуска распознавания.

Формат запроса предварительного импорта формы совпадает с форматом запросом запуска распознавания формы, за тем лишь исключением, что в контейнере сообщения передается другой код (`"MessageType": 102`).

```
{
    "MessageType": 102,
    "Data": {
        "Key": "a0bc1744-010c-4414-84e6-acabeff7b517",
        "Name": "MyApp-Form",
        "Childs": [
            {
                "DateTimeFormat": "dd.MM.yyyy",
                "Type": 4,
                "Synonyms": [
                    "Дата заполнения",
                    "Дата проведения осмотра"
                ],
                "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
            }
        ],
        "CultureCode": "ru-RU"
    }
}
```

В ответ Voice2X отправляет сообщение с описанием предварительно импортированных данных.

В случае, если на форме есть ссылочные/перечисляемые поля, рекомендуется сначала импортировать их.

Если справочники/перечисления были предварительно импортированы, тогда при импорте формы не требуется передать данные (`"AvailableValues"`). То есть вместо полного описания достаточно будет передать:

```
{
  "HandbookKey": "38d61a2b_cde8_4d17_b806_60c5fa8c8769",
  "Type": 7,
  "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
}
```

Полный пример, в котором предполагается, что справочники и перечисления импортированы ранее:

```
{
   "Key": "a0bc1744-010c-4414-84e6-acabeff7b517",
   "Name": "MyApp-Form",
   "Childs": [
      {
        "DateTimeFormat": "dd.MM.yyyy",
        "Type": 4,
        "Synonyms": [
          "Дата заполнения"
        ],
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
      },
      {
        "Type": 1,
        "Synonyms": [
          "Наличие страховки",
          "Страховка"
        ],
        "Key": "1d77096e-53eb-48ea-bfad-4ea3bab2b3a7"
      },
      {
        "Type": 2,
        "Synonyms": [
          "Высота контейнера",
          "Высота"
        ],
        "Key": "b5dcd5b9-576a-420b-aea2-b81171a2ee8e"
      },
      {
        "Type": 3,
        "Synonyms": [
          "Концентрация раствора",
          "Высота"
        ],
        "Key": "93d60cc9-fa05-4cdc-b9dd-628e95fe51d9"
      },
      {
        "HandbookKey": "38d61a2b_cde8_4d17_b806_60c5fa8c8769",
        "AvailableValues": [
        {
            "Key": "0",
            "Value": "Незначительная",
            "Synonyms": [
                "Слабовязкая"
            ]
        },
        {
            "Key": "1",
            "Value": "Малая",
            "Synonyms": [
                "Маловязкая"
            ]
        },
        {
            "Key": "2",
            "Value": "Повышенная",
            "Synonyms": [
                "Вязкая"
            ]
        },
        {
            "Key": "3",
            "Value": "Высокая",
            "Synonyms": [
                "Высоковязкая"
            ]
        }
          ],
        "Type": 7,
        "Synonyms": [
          "Величина вязкости"
        ],
        "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
      },
      {
        "Type": 0,
        "Synonyms": [
          "Описание"
        ],
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
      }
   ],
   "CultureCode": "ru-RU"
}
```

MessageType 103. Запрос списка данных по типу

В результате обработки запроса Voice2X возвращает структуру, которая содержит список ранее импортированных данных указанного типа, для каждого из которых определены ключ-идентификатор справочника и HashCode.

HashCode — это хэш данных, вычисляемый на стороне Voice2X. Служит для поддержки версионности описания формы. HashCode вычисляется на основании JSON-а с описанием формы по алгоритму SHA256. В вычислении не учитываются поля `ThemeId` и данные справочников (при условии, что справочники внешние, то есть содержат `HandbookKey`)

Поле `Data` содержит тип запрашиваемых данных, возможные типы:

- **0** — справочник;
- **1** — форма;
- **2** — набор команд.

Если поле `Data` в сообщении не указано, то возвращается список импортированных форм.

```
{
   "MessageType": 103,
   "Data": 0
}
```

В ответ Voice2X отправляет сообщение со списком импортированных данных.

MessageType 104. Запрос данных по ключу

В результате обработки запроса Voice2X возвращает структуру, которая содержит описание предварительно импортированных данных, соответствующих переданному в запросе типу и ключу.

Поле `Data` сообщения содержит структуру, состоящую из двух полей:

Запрос на получение метаданных			
Поле	**Тип поля**	**Обязательное**	**Описание**
Key	String	Да	Ключ-идентификатор метаданных
Type	Int32	Да	Тип кешированных данных: **0** – справочник; **1** – форма; **2** – набор команд.

```
{
   "MessageType": 104,
   "Data":{
      "Type":0,
      "Key":"3b00e940-1f75-4bae-ae33-7796dbceb768"
   }
}
```

Здесь должен быть указан существующий ключ импортированных ранее данных.

В ответ Voice2X отправляет сообщение с описанием ранее импортированных данных, соответствующих переданному в запросе ключу.

MessageType 105. Установка активного элемента (группы)

ИС должна отправить в Voice2X уведомление при ручной смене пользователем фокуса на форме (на любом элементе, вкладке или группе).

Поле `Data` должно содержать идентификатор предварительно импортированного элемента формы.

```
{
   "MessageType": 105,
   "Data":"ea0ac1b8-d91d-4b98-8d63-4f565048ab77"
}
```

Передаваемый ключ должен существовать на форме.

При необходимости сбросить активную группу нужно в поле `Data` передать `null`.

```
{
   "MessageType": 105,
   "Data":null
}
```

В ответ Voice2X отправляет уведомление о фокусе на группе или элементе, если при запуске распознавания установлено свойство `CyclicFocusNotificationEnabled = true`.

MessageType 106. Запуск распознавания

Команда служит для запуска распознавания формы, описание которой передано вместе в командой или импортировано заранее.

Если описание формы, справочников и команд были импортированы заранее

Тогда для запуска распознавания ИС должна передать лишь ключ-идентификатор заполняемой формы. Иными словами, в объекте `FormMetadata` можно исключить поле `Childs`.

Ниже представлен пример JSON-сообщения, которое ИС должна передать в Voice2X для запуска распознавания.

```
{
   "MessageType": 106,
   "Data":{
      "Key":"a0bc1744-010c-4414-84e6-acabeff7b517",
      "CultureCode":"ru-RU",
      "FormCommands": [
          { "Key": "Set1" },
          { "Key": "Set2" },
          { "Key": "Set3" }
      ]
   }
}
```

В примере предполагается, что форма a0bc1744-010c-4414-84e6-acabeff7b517 и наборы команд (Set1, Set2, Set3) были предварительно импортированы.

Если описание формы, справочников и команд заранее импортированы не были

Тогда в поле **`Data`** должно быть передано полное описание формы, справочников и команд.

В качестве примера приводится запуск распознавания формы, описание которой даётся в кратком примере.

```
{
   "Key": "a0bc1744-010c-4414-84e6-acabeff7b517",
   "Name": "MyApp-Form",
   "Childs": [
      {
        "DateTimeFormat": "dd.MM.yyyy",
        "Type": 4,
        "Synonyms": [
          "Дата заполнения"
        ],
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
      },
      {
        "Type": 1,
        "Synonyms": [
          "Наличие страховки",
          "Страховка"
        ],
        "Key": "1d77096e-53eb-48ea-bfad-4ea3bab2b3a7"
      },
      {
        "Type": 2,
        "Synonyms": [
          "Высота контейнера",
          "Высота"
        ],
        "Key": "b5dcd5b9-576a-420b-aea2-b81171a2ee8e"
      },
      {
        "Type": 3,
        "Synonyms": [
          "Концентрация раствора",
          "Высота"
        ],
        "Key": "93d60cc9-fa05-4cdc-b9dd-628e95fe51d9"
      },
      {
        "HandbookKey": "38d61a2b_cde8_4d17_b806_60c5fa8c8769",
        "AvailableValues": [
        {
            "Key": "0",
            "Value": "Незначительная",
            "Synonyms": [
                "Слабовязкая"
            ]
        },
        {
            "Key": "1",
            "Value": "Малая",
            "Synonyms": [
                "Маловязкая"
            ]
        },
        {
            "Key": "2",
            "Value": "Повышенная",
            "Synonyms": [
                "Вязкая"
            ]
        },
        {
            "Key": "3",
            "Value": "Высокая",
            "Synonyms": [
                "Высоковязкая"
            ]
        }
          ],
        "Type": 7,
        "Synonyms": [
          "Величина вязкости"
        ],
        "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
      },
      {
        "Type": 0,
        "Synonyms": [
          "Описание"
        ],
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
      }
   ],
      "CultureCode":"ru-RU",
      "FormCommands":[
          {
```

```
            "Key":"Set1",
            "Childs":[
               {
                  "Type":0,
                  "Key":"Finish",
                  "Synonyms":[
                     "Завершить"
                  ]
               }
            ]
         },
         {
            "Key":"Set2",
            "Childs":[
               {
                  "Type":0,
                  "Key":"Decline",
                  "Synonyms":[
                     "отклонить"
                  ]
               },
               {
                  "Type":0,
                  "Key":"Upload",
                  "Synonyms":[
                     "загрузить"
                  ]
               }
            ]
         }
      ]
}
```

После запуска распознавания Voice2X отправляет уведомление об изменении состояния режима распознавания и целый ряд сообщений, описанных в разделе Сообщения от Voice2X, которые ИС должна корректно обрабатывать:

- Уведомление о распознавании значения
- Уведомление о фокусе на группе (или элементе)
- Уведомление о распознавании нескольких полей значения
- Уведомление о предварительном распознавании
- Уведомление о срабатывании деактивации распознавания
- Уведомление о срабатывании команды с параметрами
- Уведомление о распознавании значения без ключа

MessageType 107. Остановка распознавания

Для остановки распознавания и завершения заполнения формы ИС должна отправить в Voice2X сообщение с командой переключения режим распознавания.

```
{
  "MessageType": 107
}
```

В ответ Voice2X отправляет уведомление об изменении текущего режима распознавания.

MessageType 110. Изменение параметров заполнения формы

Дополнительные параметры заполнения формы (FormSettings) можно менять в процессе распознавания без его перезапуска. Такой запрос следует использовать, если заполнение формы уже запущено, а настройки нужно поменять «на лету». В противном случае (распознавание не запущено) запрос будет проигнорирован.

```
{
   "MessageType":110,
   "Data":{
      "TtsFieldNameEnabled":true,
      "TtsFieldValueEnabled":false,
      "TtsTextFieldValueEnabled":true,
      "CyclicFocusNotificationEnabled":false,
      "WizardMode_TtsFullGroupNamingEnabled":false,
      "WizardMode_IsAutoSwitchingEnabled":true,
      "WizardMode_IsAutoSwitchingEnabledForText":true
   }
}
```

Установленные таким способом параметры действуют только на текущий запуск распознавания формы. Они будут сброшены при следующем запуске распознавания на значения, содержащиеся в описании формы. Чтобы форма всегда запускалась с указанными дополнительными параметрами, следует изменить их в описании формы или передавать это сообщение каждый раз во время распознавания этой формы.

Ответа от Voice2X не предусмотрено.

MessageType 111. Состояние синтезатора речи

Информационная система может явно запросить у Voice2X текущее состояние синтезатора речи.

Выполнение запроса через WebSocket может быть не очень удобным из-за того, что WS подразумевает асинхронность, а логика этого запроса больше соответствует синхронному запросу. Поэтому запросить список доступных словарей распознавания можно синхронным запросом через REST API. Однако это не является обязательным.

```
{
   "MessageType":111
}
```

В ответ от Voice2X отправляет уведомление об изменении состояния синтезатора речи.

MessageType 113. Импорт набора голосовых команд

Наравне со справочниками и описанием формы, до начала распознавания возможно импортировать в систему список собственных голосовых команд формы (FormCommandSetMetadata).

Ниже приводится пример сообщения импорта набора команд и описание полей передаваемых данных.

```
{
   "MessageType":113,
   "Data":{
      "Key":"Set3",
      "CultureCode":"ru-RU",
      "Childs":[
         {
            "Type":2,
            "Key":"GoTo",
            "Synonyms":[
               "перейти к странице"
            ]
         },
         {
            "Type":0,
            "Key":"Clear",
            "Synonyms":[
               "очистить"
            ]
         }
      ]
   }
}
```

В ответ от Voice2X отправляет сообщение с описанием импортированных данных.

MessageType 114. Изменение набора голосовых команд

Текущий набор голосовых команд формы (FormCommandSetMetadata) можно изменить без изменения самой формы и её элементов формы.

Запрос может быть выполнен только в режимах заполнения формы (то есть когда распознавание уже запущено).

Если в ходе работы один из наборов команд оказывается не нужен, то следует послать данное сообщение с оставшимся набором.

```
{
   "MessageType":114,
   "Data":[
      {
         "Key":"Set1"
      },
      {
         "Key":"Set2"
      }
   ]
}
```

В ответ Voice2X отправляет уведомление об успешной смене набора команд формы.

MessageType 119. Отключение синтезатора речи

Синтез речи может быть принудительно остановлен. Для этого информационная система должна отправить запрос на отключение синтеза речи.

```
{
   "MessageType":119
}
```

Ответа от Voice2X не предусмотрено.

MessageType 120. Включение синтезатора речи

Если синтез речи был принудительно остановлен, то для включения синтеза информационная система должна отправить запрос.

```
{
   "MessageType":120
}
```

Ответа от Voice2X не предусмотрено.

MessageType 130. Распознавание в рамках одного документа

После получения данного сообщения следующая сессия распознавания будет отнесена к указанному документу.

Заполнение документа может производиться несколькими сессиями распознавания. Работая с одним документом, пользователь может:

1. запустить распознавание речи и заполнить часть документа,
2. остановить распознавание,
3. вновь запустить распознавание речи и заполнить оставшуюся часть документа,
4. снова остановить распознавание.

Если сессия распознавания относится к определённому документу, то для корректного сбора статистики ИС должна передать **Dictation Service** его идентификатор до запуска распознавания.

> Все сессии распознавания будут отнесены к указанному документу, пока не будет получен другой идентификатор документа.
> Сообщение следует передавать перед каждой сессией распознавания, относящейся к документу.

В запросе указываются следующие данные:

Параметр	Тип	Обязательный	Описание
Id	String	Да	Идентификатор документа в ИС.

Установить идентификатор:

```
{
   "MessageType": 130,
   "Data":{
      "Id":"d62eac9e-0f65-4e1c-b1dd-cc83c507d50b"
   }
}
```

Очистить идентификатор:

```
{
   "MessageType": 130,
   "Data":{
      "Id":""
   }
}
```

Ответа от Voice2X не предусмотрено.

Сообщения от Dictation Server

Код сообщения	Наименование
2	Уведомление об изменении режима работы (аналог 200)
3	Уведомление о результатах распознавания
5	Сообщение для пользователя (всплывающие сообщения)
7	Сообщение со списком установленных на сервере лицензий
10	Сообщение со списком установленных на сервере словарей
22	Уведомление об изменении состояния синтезатора речи (TTS)
200	Уведомление об изменении режима распознавания
201	Уведомление о распознавании значения
202	Уведомление о фокусе на группе (или элементе)
203	Сообщение со списком импортированных метаданных
204	Сообщение с описанием импортированных метаданных
205	Уведомление об ошибке
206	Уведомление о распознавании нескольких полей значения
207	Сообщение со списком доступных словарей распознавания
209	Уведомление о предварительном распознавании
210	Уведомление о срабатывании команды (без параметров)
211	Уведомление об изменении состояния синтезатора речи
214	Уведомление о срабатывании команды (с параметрами)
215	Уведомление об успешной смене набора команд формы
216	Уведомление о финальном текстовом результате
221	Уведомление о событии
222	Сообщение со списком параметров синтеза речи (TTS)

При работе в режиме слитного распознавания речи и офлайн-распознавания ИС должна обрабатывать сообщения

Код сообщения	Наименование
3	Уведомление о результатах распознавания

При работе в режиме обычного заполнения формы ИС должна обрабатывать сообщения

Код сообщения	Наименование
201	Уведомление о распознавании значения
206	Уведомление о распознавании нескольких полей значения
209	Уведомление о предварительном распознавании
210	Уведомление о срабатывании команды (без параметров)
214	Уведомление о срабатывании команды (с параметрами)
216	Уведомление о финальном текстовом результате

При работе в режиме последовательного заполнения формы ИС должна обрабатывать аудио от TTS и сообщения

Код сообщения	Наименование
201	Уведомление о распознавании значения
206	Уведомление о распознавании нескольких полей значения
209	Уведомление о предварительном распознавании
210	Уведомление о срабатывании команды (без параметров)
214	Уведомление о срабатывании команды (с параметрами)

216	Уведомление о финальном текстовом результате

MessageType 2. Изменён режим работы

В ответ на запуск распознавания и остановку распознавания Voice2X отправляет уведомление об изменении состояния режима распознавания в момент его переключения.

Содержимое ответа описывается классом RecognitionModeInfo.

Фактически, при изменении режима распознавания Voice2X может отправлять сразу несколько сообщений. Так, при переключении из текущего режима в новый будут сформированы:

- сообщение о начале запуска нового режима (Mode: 3, State: 0);
- сообщение об остановке текущего режима (Mode: 1, State: 2);
- сообщение о запуске нового режима (Mode: 3, поле State: 1).

Состояние (State) принимает следующие значения:

0 — запускается;
1 — запущен и работает;
2 — отключен;
3 — отключается.

```
{
   "MessageType":2,
   "Data":{
      "Mode":3,
      "State":1,
      "Info":"a0bc1744-010c-4414-84e6-acabeff7b517"
   }
}
```

MessageType 3. Результат распознавания

Voice2X отправляет данное сообщение в процессе распознавания речи. Сообщение может содержать предварительный и финальный результат распознавания. Промежуточный результат может быть использован ИС, чтобы дать пользователю обратную связь в ходе распознавания текста, а финальный результат следует использовать для заполнения полей ИС.

Тело сообщения содержит объект класса `TextResult`.


```
{
    "MessageType": 3,
    "Data": {
        "Text": "моя команда",
        "IsFinal": true,
        "Words": [
            {
                "Begin": 90,
                "DictorNum": 0,
                "EndSentence": 0,
                "Index": 0,
                "Length": 330,
                "ProbabilityNorm": 0.6048625111579895,
                "Text": "моя",
                "PunctuationText": ""
            },
            {
                "Begin": 420,
                "DictorNum": 0,
                "EndSentence": 0,
                "Index": 0,
                "Length": 690,
                "ProbabilityNorm": 0.6048625111579895,
                "Text": "команда",
                "PunctuationText": ""
            }
        ],
        "CommandId": "Test",
        "Length": 1110
    }
}
..... 1 мс спустя .....
{
    "MessageType": 3,
    "Data": {
        "Text": "остановить",
        "IsFinal": false,
        "Words": [
            {
                "Begin": 1560,
                "DictorNum": 0,
                "EndSentence": 0,
                "Index": 0,
                "Length": 690,
                "ProbabilityNorm": 0.0,
                "Text": "остановить",
                "PunctuationText": ""
            }
        ],
        "CommandId": "",
        "Length": 2250
    }
}
..... 1 мс спустя .....
{
    "MessageType": 3,
    "Data": {
        "Text": "остановить распознавание",
        "IsFinal": false,
        "Words": [
            {
                "Begin": 1560,
                "DictorNum": 0,
                "EndSentence": 0,
                "Index": 0,
                "Length": 690,
                "ProbabilityNorm": 0.0,
                "Text": "остановить",
                "PunctuationText": ""
            },
```

```
            {
                "Begin": 2250,
                "DictorNum": 0,
                "EndSentence": 0,
                "Index": 0,
                "Length": 840,
                "ProbabilityNorm": 0.0,
                "Text": "распознавание",
                "PunctuationText": ""
            }
        ],
        "CommandId": "",
        "Length": 3090
    }
}
```

MessageType 5. Сообщение пользователю

Voice2X отправляет в ИС дополнительные сообщения, которые может быть необходимо отображать пользователю.

Тело сообщения содержит объект класса `UserNotification`.

Пример ответа:

```
{
    "MessageType": 5,
    "Data": {
        "Message": "Не передан AccessToken",
        "Type": 1,
        "Action": 0
    }
}
```

MessageType 7. Список установленных на сервере лицензий

После установления WS-соединения Voice2X отправляет сообщение со списком установленных лицензий.

Возвращаемое значение описывается классом `ProtectionInfo`.


```
{
    "MessageType": 7,
    "Data": {
        "ProtectionData": {
            "LastUpdateTimestamp": "2023-06-22T09:53:44.4686275+03:00",
            "CountOfCapturedLicenses": 0,
            "CountOfLicenses": 0,
            "DaysLeft": -1,
            "LaunchCount": 0,
            "LicenseType": 2,
            "Type": 1
        },
        "Type": 0,
        "ComponentId": "common_7.1.18"
    }
}
```

MessageType 10. Список установленных на сервере языковых моделей

После установления WS-соединения Voice2X отправляет сообщение со списком установленных специальных языковых моделей.

Содержимое ответа описывается классом ServerThemesInfo.


```
{
    "MessageType": 10,
    "Data": {
        "ThemeInfos": [
            {
                "Id": "common_7.1.18:eos",
                "ThemeId": "common",
                "Name": "Базовый",
                "Version": "7.1.18",
                "ShopUrl": "",
                "NewVersionInfoUrl": "",
                "CompositionType": 0,
                "LicenseState": 0,
                "Description": "Состав:\n
        - Общий словарь русского языка"
            }
        ]
    }
}
```

MessageType 22. Список параметров синтезатора речи

Voice2X отправляет данное сообщение в ответ на запрос настроек синтезатора речи.

Возвращаемые значения описываются классом `TtsStateInfo`.

```
{
    "MessageType": 22,
    "Data": {
        "VoiceList": [
            {
                "Name": "Alexander"
            },
            {
                "Name": "Julia"
            }
        ],
        "SelectedVoice": {
            "Name": "Alexander"
        },
        "Speed": 1.4,
        "State": 0
    }
}
```

MessageType 200. Изменено состояние режима распознавания

В ответ на запуск распознавания и остановку распознавания Voice2X отправляет уведомление об изменении состояния режима распознавания в момент его переключения.

Содержимое ответа описывается классом RecognitionModeInfo.

Фактически, при изменении режима распознавания Voice2X может отправлять сразу несколько сообщений. Так, при переключении из текущего режима в новый будут сформированы:

- сообщение о начале запуска нового режима (Mode: 3, State: 0);
- сообщение об остановке текущего режима (Mode: 1, State: 2);
- сообщение о запуске нового режима (Mode: 3, поле State: 1).

Состояние принимает следующие значения:

0 — запускается;
1 — запущен и работает;
2 — отключен;
3 — отключается.

```
{
   "MessageType":200,
   "Data":{
      "Mode":3,
      "State":1,
      "Info":"a0bc1744-010c-4414-84e6-acabeff7b517"
   }
}
```

MessageType 201. Распознано значение

Voice2X отправляет данное сообщение, если распознанное значение удалось сопоставить какому-то элементу формы.

Возвращаемое значение описывается классом `ValuedElementInfo<T>`, который имеет специфичную реализацию для каждого типа поля.

В случае возвращения данных для текстового поля Voice2X отправит следующее сообщение:

Для заполнения поля «Наличие страховки» значением *есть* из примера это будет сообщение:

```
{
    "MessageType": 201,
    "Data": {
        "Type": 1,
        "Value": true,
        "Childs": null,
        "HandbookRow": null,
        "Key": "1d77096e-53eb-48ea-bfad-4ea3bab2b3a7"
    }
}
```

MessageType 202. Фокус на группе или элементе

Voice2X отправляет данное сообщение:

- когда ИС установила активный элемент при включённом свойстве `CyclicFocusNotificationEnabled = true`.
- когда пользователь произносит название какой-либо группы и эта группа становится активной (см. описание ContainerMetadata).

Ответ содержит ключ элемента, на который ИС должна перевести фокус.

Для перехода к полю «Концентрация раствора» из примера это будет сообщение:

```
{
   "MessageType": 202,
   "Data":{
      "Key":"93d60cc9-fa05-4cdc-b9dd-628e95fe51d9"
   }
}
```

MessageType 203. Список импортированных данных

Voice2X отправляет данное сообщение в ответ на запрос списка данных по типу.

Ответ содержит список предварительно импортированных данных для каждого из которых определены тип, ключ-идентификатор справочника и HashCode.

Для запроса всех справочников формы из примера это будет:


```
{
    "MessageType": 203,
    "Data": [
        {
            "Key": "3b00e940-1f75-4bae-ae33-7796dbceb768",
            "Type": 0,
            "HashCode": "*??\u000F?????W]>???????\u0015S???\u000F\u0005K??\u0004?9"
        },
        {
            "Key": "8ecafe5b-a6cf-4576-b423-cd915d25bba3",
            "Type": 0,
            "HashCode": "*??\u000F?????W]>???????\u0015S???\u000F\u0005K??\u0004?9"
        },
        {
            "Key": "38d61a2b-cde8-4d17-b806-60c5fa8c8769",
            "Type": 0,
            "HashCode": "??(?(?\u0018\u0018\u0002?tRE%??l?\u0013_????\u0019???\u0014w\u0016"
        }
    ]
}
```

MessageType 204. Описание импортированных данных

Voice2X отправляет данное сообщение при импорте формы, справочника или команд, а также при запросе импортированных данных по ключу.

Содержит описание ранее импортированных метаданных, соответствующих переданному в запросе ключу.

Для запроса формы из примера по её ключу это будет:


```
{
    "MessageType": 204,
    "Data": {
        "Key": "3b00e940-1f75-4bae-ae33-7796dbceb768",
        "Type": 0,
        "HashCode": "*??\u000f?????W]>???????\u0015S???\u000f\u0005K??\u0004?9"
    }
}
```

MessageType 205. Ошибка

Voice2X отправляет уведомление в момент возникновения ошибки. Содержимое описывается классом `ErrorInfo`.

Например, попытка загрузки некорректного описания формы (описание формы) вернёт ошибку.

Обработанная ошибка:

```
{
    "MessageType": 205,
    "Data": {
        "Text": "В сохраняемой форме не задан ключ."
    }
}
```

Необработанная ошибка (не удалось обработать структуру JSON-файла):

```
{
    "MessageType": 5,
    "Data": {
        "Text": "Внутренняя ошибка сервера, попробуйте выполнить операцию позже",
        "DebugText": "Newtonsoft.Json.JsonReaderException: Invalid property identifier character:
                     {. Path 'Data.Childs[0]', line 7, position 6.\r\n
                     at Newtonsoft.Json.JsonTextReader.ParseProperty()\r\n
                     at Newtonsoft.Json.JsonTextReader.ParseObject()\r\n
                     at Newtonsoft.Json.JsonWriter.WriteToken(JsonReader reader, Boolean writeChildren,
                     Boolean writeDateConstructorAsDate, Boolean writeComments)\r\n
                     at
Newtonsoft.Json.Serialization.JsonSerializerInternalReader.CreateJObject(JsonReader reader)\r\n
                     at Newtonsoft.Json.Serialization.JsonSerializerInternalReader.CreateObject(JsonReader
reader,
                     Type objectType, JsonContract contract, JsonProperty member, JsonContainerContract
containerContract,
                     JsonProperty containerMember, Object existingValue)\r\n
                     at
Newtonsoft.Json.Serialization.JsonSerializerInternalReader.CreateValueInternal(JsonReader reader,
                     Type objectType, JsonContract contract, JsonProperty member, JsonContainerContract
containerContract,
                     JsonProperty containerMember, Object existingValue)\r\n
                     at
Newtonsoft.Json.Serialization.JsonSerializerInternalReader.ResolvePropertyAndCreatorValues(
                     JsonObjectContract contract, JsonProperty containerProperty, JsonReader reader, Type
objectType)\r\n
                     at
Newtonsoft.Json.Serialization.JsonSerializerInternalReader.CreateObjectUsingCreatorWithParameters(
                     JsonReader reader, JsonObjectContract contract, JsonProperty containerProperty,
                     ObjectConstructor`1 creator, String id)\r\n
                     at Newtonsoft.Json.Serialization.JsonSerializerInternalReader.CreateObject(JsonReader
reader,
                     Type objectType, JsonContract contract, JsonProperty member, JsonContainerContract
containerContract,
                     JsonProperty containerMember, Object existingValue)\r\n
                     at
Newtonsoft.Json.Serialization.JsonSerializerInternalReader.CreateValueInternal(JsonReader reader,
                     Type objectType, JsonContract contract, JsonProperty member, JsonContainerContract
containerContract,
                     JsonProperty containerMember, Object existingValue)\r\n
                     at Newtonsoft.Json.Serialization.JsonSerializerInternalReader.Deserialize(JsonReader
reader,
                     Type objectType, Boolean checkAdditionalContent)\r\n
                     at Newtonsoft.Json.JsonSerializer.DeserializeInternal(JsonReader reader, Type
objectType)\r\n
                     at Newtonsoft.Json.JsonConvert.DeserializeObject(String value, Type type,
JsonSerializerSettings settings)\r\n
                     at Newtonsoft.Json.JsonConvert.DeserializeObject[T](String value,
JsonSerializerSettings settings)\r\n
                     at Dictation.Common.Connection.MessageWrapper.Deserialize[T](String _text)\r\n
                     at Dictation.Common.Connection.MessageTransceiver`1.ProcessMessage(String _message)"
    }
}
```

MessageType 206. Найдено несколько полей для распознанного значения

Voice2X отправляет данное сообщение в случае, если распознанному значению соответствует несколько полей. Обычно это говорит о том, что на форме на одном общем уровне вложенности существуют два и более поля с одинаковыми синонимами (названиями). Информационная система должна сама решать, что делать с такими данными: какое именно поле при это заполнять. В общем случае разработчиками ИС рекомендуется не допускать совпадения синонимов на одном уровне вложения.

Возвращаемые значения описываются классом `ValuedElementInfo<T>`, который имеет специфичную реализацию для каждого типа поля.

```
{
    "MessageType": 206,
    "Data": [
        [
            {
                "Type": 4,
                "Value": "20.12.2018",
                "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
            }
        ],
        [
            {
                "Type": 4,
                "Value": "20.12.2018",
                "Key": "9707c7c9-5cc3-441b-8418-a840b8bfb356"
            }
        ]
    ]
}
```

MessageType 207. Список доступных языковых моделей

Возвращаемое значение описывается классом ThemeInfo.

```
{
    "MessageType": 207,
    "Data": [
    {
        "Id": "common_7.1.18:eos",
        "ThemeId": "common",
        "Name": "Базовый",
        "Version": "6.3.2",
        "ShopUrl": "",
        "NewVersionInfoUrl": "",
        "CompositionType": 0,
        "LicenseState": 0,
        "Description": "Состав:\n\t- Общий словарь русского языка"
    },
    {
        ... //дополнительные специальные языковые модели
    }
  ]
}
```

MessageType 209. Предварительно распознана речь

Voice2X отправляет данное сообщение в процессе распознавания речи. Сообщение содержит предварительный результат распознавания: этот текст не является итоговым значением, которое можно использовать для заполнения каких-либо полей. Его назначение — отобразить динамику распознавания.

После того, как значение будет полностью произнесено, Voice2X отправляет сообщение о распознанном значении или сообщение о распознанном значении, для которого на форме не обнаружено поле.

Например, если фраза достаточно длинная, пользователю будет удобнее видеть промежуточный результат (возможно в tooltip или каком-то отдельном элементе на форме), чтобы понимать что процесс распознавания его речи активен.

В процессе заполнения поля «Описание» из примера текстовым значением будут формироваться сообщения:

```
{
    "MessageType": 209,
    "Data": "Любой произвольные темы"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текста"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст который"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст который необходимо"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст который необходимо ввести"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст который необходимо ввести в"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст который необходимо ввести в этом"
}
{
    "MessageType": 209,
    "Data": "Любой произвольной текст который необходимо ввести в это поле"
}
```

MessageType 210. Распознана команда (без параметров)

После того, как в речи пользователя распознана голосовая команда, Voice2X отправляет в ИС данное сообщение в паре с сообщением о распознавании команды. В отличие от сообщения о распознавании команды, это сообщение не может содержать дополнительные параметры.

В поле `Data` сообщение содержит идентификатор команды (см. описание класса `FormCommandSetMetadata`).

При получении такого уведомления ожидается, что ИС выполнит соответствующее команде действие.

```
{
    "MessageType": 210,
    "Data": "Upload"
}
```

MessageType 211. Изменено состояние синтезатора речи

TTS (Text To Speech) — служба Voice2X, отвечающий за синтез речи. TTS синтезирует звук голосового подтверждения распознанных данных и озвучивает поле для заполнения в режиме последовательного заполнения формы.

Voice2X отправляет данное сообщение в ответ на явный запрос состояния синтезатора или при изменении состояния модуля.

Возвращаемые значения описываются классом `TtsStateInfo`.

Класс состояния синтезатора речи			
Поле	**Тип поля**	**Обязательное**	**Описание**
VoiceList	`KeyValuePair<string, string>`	Нет	Список голосов
SelectedVoice	`KeyValuePair<string, string>`	Нет	Выбранный голос
Speed	`Decimal`	Да	Скорость воспроизведения. Допустимые значения от 0,5 до 2.
State	`Int32`	Да	Статус: **0** – всё работает нормально; **1** – внутренняя ошибка, возможно некорректно установлен синтезатор; **2** – ошибка проверки лицензии

Пример:

```
{
    "MessageType": 211,
    "Data": {
        "VoiceList": [
            {
                "Name": "Александр"
            },
            {
                "Name": "Анна"
            },
            {
                "Name": "Юлия"
            }
        ],
        "SelectedVoice": {
            "Name": "Анна"
        },
        "Speed": 1.0,
        "State": 0
    }
}
```

MessageType 214. Распознана команда

После того, как в речи пользователя распознана голосовая команда, Voice2X отправляет в ИС данное сообщение в паре с сообщением о распознавании команды (без параметров). В отличие от сообщения о распознавании команды без параметра, может содержать дополнительные параметры в поле `Value`.

Сообщение содержит распознанное значение `ValuedElementInfo<T>`, аналогично сообщению с кодом 1.

При получении такого уведомления ожидается, что ИС выполнит соответствующее команде действие.

Если команда присылает параметр:

```
{
    "MessageType": 214,
    "Data": {
        "Type": 2,
        "Value": 7,
        "Key": "GoTo"
    }
}
```

Если команда не присылает параметр:

```
{
    "MessageType": 14,
    "Data": {
        "Type": 0,
        "Key": "Upload"
    }
}
```

MessageType 215. Изменён набор команд

Voice2X отправляет данное сообщение при изменении набора голосовых команд.

Ответ содержит список ранее импортированных наборов команд для каждого из которых определены тип, ключ-идентификатор и HashCode.


```
{
    "MessageType": 215,
    "Data": [
        {
            "Key": "Set1",
            "Type": 2,
            "HashCode": "'9?l??`?????D??2+'?--?e'?J? d??v"
        },
        {
            "Key": "Set2",
            "Type": 2,
            "HashCode": "????X?\u0006??????濘?\u0005??oXb:?F???|o"
        },
        {
            "Key": "Set3",
            "Type": 2,
            "HashCode": "\u0007xbz??\u000BKI??J?s??¯??N\u001B\u0017?*???⁄\u001B"
        }
    ]
}
```

MessageType 216. Распознано значение без ключа

Voice2X отправляет данное сообщение при успешном распознавании значения, для которого на форме отсутствует поле.

ИС должна самостоятельно принять решение, что делать с такими значениями.

```
{
    "MessageType": 216,
    "Data": "Мой дядя самых честных правил"
}
```

MessageType 221. Системное событие

Voice2X отправляет данное сообщение при подключении или отключении сервиса распознавания.

Содержимое описывается классом `NotificationInfo`.

```
{
    "MessageType": 221,
    "Data": {
        "Code": 2
    }
}
```

Передача аудиоданных в Dictation Server

Звуковые данные передаются по websocket в пакетах типа `binary`.

На стороне клиентского приложения звук должен быть захвачен в формате PCM-16, с частотой дискретизации 16000 Гц (mono PCM signed 16-bit little-endian).

Рекомендуется передавать звук пакетами по 200 мс (т. е. по 6400 байт) сразу, по мере формирования, не накапливая пакеты.

В ответ от Voice2X отправляет сообщение с результатом распознавания.

Получение аудиоданных от синтезатора речи (TTS)

Если в параметрах формы (`FormSettings`) содержатся указания к активации синтезатора речи для озвучивания названия полей и/или значений полей, то вашему приложению потребуется самостоятельно получать и воспроизводить аудиоданные от синтезатора речи.

Звуковые данные передаются клиентскому приложению по websocket в пакетах типа binary.

Сервер синтезирует звук (TTS) в формате PCM-16, с частотой дискретизации 22050 Гц (mono PCM signed 16-bit little-endian).

При работе через Postman пакет бинарных данных получается в кодировке Base64. Возможно, вам потребуется преобразовать его в hex для последующего использования.

Обработка ошибок

Сообщения об ошибке в WebSocket

В момент возникновения ошибки Voice2X отправляет уведомление, описываемое классом `ErrorInfo`.

Класс описания возникшей ошибки			
Поле	**Тип поля**	**Обязательное**	**Описание**
Text	`String`	Да	Текстовое сообщение об ошибке
DebugText	`String`	Нет	Расширенное описание ошибки

Например, попытка загрузки некорректного описания формы вернёт ошибки.

```
{
  "MessageType": 5,
  "Data": {
  "Text": "Некорректный формат загружаемых метаданных"
  }
}
{
    "MessageType": 205,
    "Data": {
        "Text": "Внутренняя ошибка сервера, попробуйте выполнить операцию позже",
        "DebugText": "Newtonsoft.Json.JsonReaderException: JsonToken EndObject is not valid for closing
JsonType Array."
    }
}
```

Сообщения об ошибке в Rest (http)

Rest API вернёт ошибку, если с вашими вызовами что-то пойдет не так. Каждая ошибка имеет http-код, а также возвращает тип ошибки и краткое объяснение при возникновении ошибки.

При использовании любого API помните, что ошибки и исключения (например, проблемы с подключением к серверу или простои) редко, но все же случаются. Внимательно следите за ошибками.

Причины ошибок:

- Коды **4xx** указывают на ошибку в запросе. Если вы получили в ответ ошибку 4xx, приведенный ниже глоссарий ошибок поможет понять причину неполадки.
- Коды **5xx** указывают на проблему со стороны Voice2X. Если вы получаете в ответ ошибку 5xx, то рекомендуем проверить, не возникли ли какие-либо общесистемные проблемы в Voice2X.

Глоссарий распространённых ошибок http

Ошибка	Описание
400	Такой ответ можно получить, если Voice2X не смог правильно отреагировать на ваш запрос. Например, нарушены правила синтаксиса. Повторный запрос не следует отправлять до тех пор, пока ошибка не будет исправлена. Некоторые из наиболее частые причины: **Bad Request** (Неверный запрос). Общий тип ошибки, связанный с невозможностью обработки запроса, и обычно в сообщении содержится пояснение. **Invalid Resource** (Недопустимый ресурс). Отправленный текст POST не прошел нашу проверку ввода. Эта ошибка может включать дополнительное свойство «ошибки» со списком проблем проверки. **Invalid Action** (Недопустимое действие). Запрошенное действие недействительно для вызываемого ресурса. Возвращается, когда вы пытаетесь получить доступ к действию для ресурса, который не поддерживает это действие. **JSON Parse Exception** (Исключение анализа JSON). JSON, отправленный в тексте запроса, не является допустимым JSON.
401	**Unauthorized** (Не авторизовано). Voice2X не может обработать запрос без аутентификации. Нужно добавить в заголовок запроса поле Authorization или проверить правильность данных в этом поле, если оно уже присутствует.
403	**Forbidden** (Запрещено) Voice2X отказывается обработать запрос, потому что у вас нет прав на просмотр содержимого: не имеете доступа, либо ваш уровень не разрешает доступ к конечной точке.
404	**Not Found** (Ресурс не найден). Вы пытаетесь запросить ресурс, который не существует. Например, случайно ошиблись при вводе узлов.
405	**Method Not Allowed** (Метод запрещён). Указанный в запросе метод нельзя использовать. Ошибка обычно возникает из-за того, что конечная точка не поддерживает HTTP-метод запроса. Для решения проблемы узнайте, какие методы разрешены для каждого ресурса.
414	**URI Too Long** (Слишком длинный URI). URI превышает максимально допустимую длину или имеет неверный формат.
422	**Unprocessable Entity** (Необрабатываемый запрос). Синтаксис запроса правильный, но из-за логической ошибки сервер не может его выполнить.
500	**Internal Server Error** (Внутренняя ошибка сервера). Во время обработки запроса произошла внутренняя ошибка. Voice2X столкнулся с ситуацией, которую он не знает, как обработать.

Ограничения

Ключи и идентификаторы полей и форм не должны содержать символы, которые запрещены к использованию в именах файлов и коды ANSII ниже 0x20. В частности, запрещено использовать:

- < (меньше чем);
- > (больше чем);
- : (двоеточие);
- " (двойная кавычка);
- / (косая черта);
- \ (обратная косая черта);
- | (вертикальная полоса или канал);
- ? (вопросительный знак);
- * (звездочка).

Синонимы и названия полей не должны начинаться со слов, совпадающих с возможными значениями булевых полей:

- да;
- нет;
- отсутствует;
- отсутствуют;
- есть.

Если без этих слов не обойтись, перенесите их в середину или конец имени поля. Иначе поля будут неверно заполняться при диктовке.

Возвращаемые данные при заполнении полей

После успешного распознавания Voice2X возвращает значение поля в виде класса `ValuedElementInfo<T>`, обёрнутого в контейнер `DictationApiMessage<T>`.

```
{
    "MessageType": 1,
    "Data": {
        "Type": 7,
        "Value": "Какой-то произвольный текст",
        "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
    }
}
```

В зависимости от распознанного типа поля, `ValuedElementInfo<T>` принимает следующий вид:

- ValuedElementInfo<String>
- ValuedElementInfo<Boolean>
- ValuedElementInfo<Int32>
- ValuedElementInfo<Decimal>
- ValuedElementInfo<KeyValuePair<string, string>[]>

ValuedElementInfo<String>

ValuedElementInfo<String>			
Класс содержит информацию о значении поля в текстовом виде			
Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Ключ-идентификатор элемента на форме.
Type	`Int32`	Да	Равно: `'0'` – для текстового поля, `'4'` – для поля типа дата, `'5'` – для поля типа время, `'6'` – для поля типа дата и время, `'8'` – для поля типа промежуток времени,
Value	`String`	Да	Распознанное значение

ValuedElementInfo<Boolean>

ValuedElementInfo<Boolean>			
Класс содержит информацию о значении поля в булевом виде			
Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Ключ-идентификатор элемента на форме.
Type	`Int32`	Да	Всегда равно `'1'`
Value	`Boolean`	Да	Распознанное значение `'true'` или `'false'`

ValuedElementInfo<Int32>

ValuedElementInfo<Int32>			
Класс содержит информацию о значении поля в целочисленном виде			
Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Ключ-идентификатор элемента на форме.
Type	`Int32`	Да	Всегда равно `'2'`
Value	`Int32`	Да	Распознанное значение

ValuedElementInfo<Decimal>

ValuedElementInfo<Decimal>			
Класс содержит информацию о значении поля в виде числа с плавающей точкой			
Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Ключ-идентификатор элемента на форме.
Type	`Int32`	Да	Всегда равно `'3'`
Value	`Decimal`	Да	Распознанное значение

ValuedElementInfo<KeyValuePair<string, string>[]>

ValuedElementInfo<KeyValuePair<string, string>[]>			
Класс содержит информацию о выбранном значении из списка (справочника)			
Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Ключ-идентификатор элемента на форме
Type	`Int32`	Да	Всегда равно `'7'`
HandbookRow	`KeyValuePair<string,string>`	Да, если нет `Value`	Распознанное значение перечисления
Value	`String`	Да, если нет `HandbookRow`	Распознано произвольное значение *Поле может вернуться значение без ключа, если пользователь применил раздельный ввод (выбрал текущий ComboBox), и при этом распознался простой текст (без привязки к ключу или полю).*

Формат описания формы

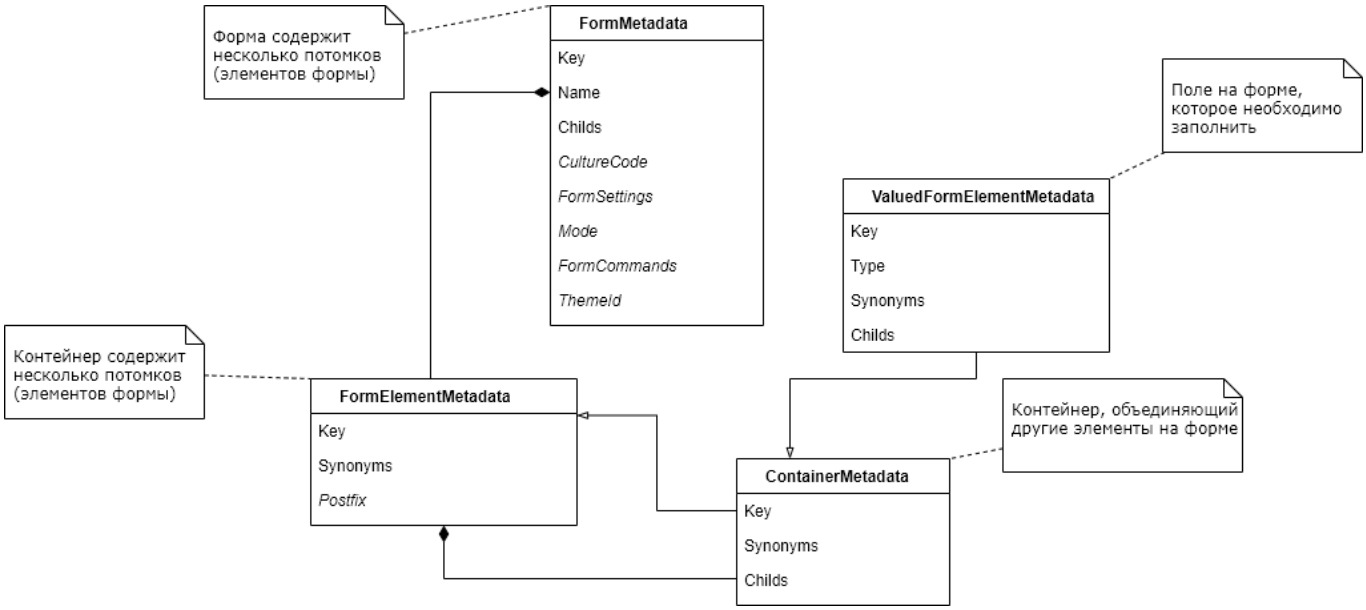
Заполняемая форма описывается объектом `FormMetadata`.

При передаче этот объект вкладывается в контейнер `DictationApiMessage<T>`. В зависимости от желаемого действия, полю `MessageType` контейнера `DictationApiMessage<T>` может быть установлено значение, которое позволяет:

- произвести предварительную загрузку описания формы без запуска распознавания,
- загрузить описание формы и запустить распознавание (при этом произойдёт смена режима работы).

Описание формы необходимо передать в формате JSON. В случае невозможности проинициализировать форму (например, из-за ошибки при формировании метаданных) вернётся сообщение об ошибке.

Диаграмма классов показывает базовую абстрактную структуру отправляемых данных.



Типы данных для заполнения полей

Заполняемая форма может содержать поля со следующими типами данных:

Код	Тип данных	Назначение	В JSON
0	String	Простые текстовые данные	Описание
1	Boolean	Логическое значение (да/нет)	Описание
2	Int32	Целое число Минимальное значение: -2 147 483 648; Максимальное значение: 2 147 483 647	Описание
3	Decimal	Десятичная дробь	Описание
4	Date	Дата	Описание
5	Time	Время	Описание
6	DateTime	Дата и время	Описание
7	Link	Перечисление или ссылка на справочник на стороне внешнего клиента (ИС)	Описание
8	TimeSpan	Промежуток времени. Примеры: “Один год” “Пять лет” “Четыре года и семь месяцев” “Двенадцать дней” “Восемь часов”	Описание
9	–..–	Зарезервировано	

При описании каждого типа данных должен использоваться подходящий класс, унаследованный от `ValuedFormElementMetadata`.

Указанные типы могут быть сгруппированы при помощи класса `ContainerMetadata`.

Классы структуры формы

При формировании описания формы формируется объект на основе следующих классов:

Класс FormMetadata — контейнер для:

- описания формы в виде дочерних элементов FormElementMetadata, содержащих группы заполняемых полей и сами поля,
- используемого при заполнении словаря,
- параметров её заполнения FormSettings,
- режима работы Voice2X,
- голосовых команд FormCommandSetMetadata.

Класс FormMetadata

Класс содержит полное описание формы			
Поле	**Тип поля**	**Обязательное**	**Описание**
Key	`String`	Да	Идентификатор формы. Используется для предварительного импорта формы и при запуске её распознавания
Name	`String`	Нет	Название формы
Childs	Список из дочерних элементов класса `FormElementMetadata`	Да	Коллекция (иерархия) элементов на форме (поля формы)
CultureCode	`String`	Нет	Код культуры в соответствии с которым будут сериализованы распознанные данные. Например: “ru-RU”. Если не передавать, будет использован InvariantCulture
FormSettings	`FormSettings`	Нет	Дополнительные параметры распознавания
Mode	`Int32` (nullable)	Нет	Режим заполнения формы. **0** – Свободный в формате ключ + значение (используется по умолчанию). **1** – Последовательный. Система произносит (TTS) ключ и ожидает от пользователя произнесения значения
FormCommands	`List <FormCommandSetMetadata>`	Нет	Пользовательские команды для управления формой
ThemeId	`String`	Да	Идентификатор специальной языковой модели, которая должна использовать при обработке полей, поддерживающих свободный ввод

Выбор специальной языковой модели распознавания речи влияет **только** на заполнение **полей со свободным вводом**: строковые поля и комбобоксы с возможностью свободного ввода (по умолчанию эта возможность отключена). Важно понимать, что даже в этом случае влияние ограничено самим значением свободного ввода.

Например, у нас есть поле «Комментарий»:

1. *Пользователь произносит: «Комментарий» и фокус устанавливается на этом поле (модель распознавания **не оказывает влияния**).*
2. *Пользователь произносит: «В этом поле будет какой-то произвольный текст» (результат распознавания **зависит** от выбранной модели распознавания).*

Для остальных полей модель распознавания никак не влияет на их заполнение.

Пример использования:

```
{
    "Key":"a0bc1744-010c-4414-84e6-acabeff7b517",
    "Name":"Форма №1",
    "Childs":[…],
    "CultureCode":"ru-RU",
    "Mode": 1,
    "FormSettings": {…},
    "ThemeId": "main_2.3.13",
    "FormCommands": []
}
```

Класс FormElementMetadata

Абстрактный класс Конкретные реализации: ContainerMetadata и ValuedFormElementMetadata Описание элемента, находящегося на форме			
Поле	**Тип поля**	**Обязательное**	**Описание**
Key	`String`	Да	Идентификатор, который используется для предварительного импорта справочника и использовании импортированного справочника
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля/группы. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю. Например, во фразе «продольный размер 35 миллиметров» слово «миллиметров» необязательно к произношению и не влияет на итоговое значение поля (35). Но, скорее всего, пользователь произнесёт это слово. Поле не используется для строкового типа и контейнера

Класс ContainerMetadata

Элементы на заполняемой форме могут быть сгруппированы различным образом. В общем смысле под группами могут пониматься:

- группы,
- панели,
- вкладки (tab),
- иные подобные элементы.

При описании формы группа описывается классом-контейнером `ContainerMetadata`

Класс-контейнер для других элементов			
Поле	**Тип поля**	**Обязательное**	**Описание**
Childs	Список из FormElementMetadata	Да	Коллекция (иерархия) элементов на форме
Key	`String`	Да	Идентификатор контейнера в ИС
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения названия группы. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю

Пример использования контейнера:

```
{
    "Childs":[
        {…}
        {…}
    ],
    "Synonyms":[
        "Название группы",
        "Группа …"
    ],
    "Key":"ea0ac1b8-d91d-4b98-8d63-4f565048ab77"
}
```

В разных группах могут содержаться элементы с одинаковыми названиями.

Например, в группах «Без нагрузки» и «Под нагрузкой» может быть одинаковое поле «Обороты».

Для удобства заполнения подобных элементов была введена концепция «активной группы» (или поля). Изначально активная группа является пустой.

Если рассматривать тот же пример, то при произнесении пользователем фразы «Без нагрузки» соответствующая группа будет установлена как активная. Это означает, что если дальше пользователь произнесет фразу «Обороты 68», то в ИС будет отправлено уведомление о заполнении соответствующего поля в данной группе. Поле в соседней группе «Под нагрузки» будет проигнорировано.

Чтобы сделать активной другую группу, следует произнести её название (в данном случае: «Под нагрузки»).

Чтобы сделать группу неактивной, следует перейти к заполнению поля, находящегося вне группировки (в данном случае это могут быть «Дата заполнения» или «Описание»).

Таким образом, для голосового заполнения пользователю доступны все поля его текущей группы и все поля, находящиеся в иерархии выше текущей группы.

Для заполнения полей из соседних или дочерних групп необходимо будет произнести название этих групп, сделав их активными.

Ответ от Voice2X при фокусе на контейнере

Voice2X отправляет в ИС сообщение, когда пользователь произносит название какой-либо группы и эта группа становится активной.

```
{
  "MessageType": 2,
  "Data": {
      "Key": "ea0ac1b8-d91d-4b98-8d63-4f565048ab77"
  }
}
```

Обратное уведомление о фокусе на группе или элементе

ИС должна отправить в Voice2X уведомление в случае, если пользователь вручную установил фокус на любой элемент или переключился на другую группу (например, если пользователь переключился на другую вкладку на форме).

```
{
  "MessageType": 5,
  "Data": {
       "Key": "ea0ac1b8-d91d-4b98-8d63-4f565048ab77"
  }
}
```

При необходимости сбросить активную группу нужно в поле `Key` передать `null`.

```
{
  "MessageType": 5,
  "Data": {
       "Key": null
  }
}
```

Алгоритм поиска ключа в дереве формы

При голосовом заполнении сложных форм возникает проблема ввода значений полей, вложенных в группы (отдельные вкладки на форме или иные подобные элементы группировки). Проблема усугубляется, если поля имеют одинаковые названия (синонимы). Такие поля **обязательно** должны находиться в разных группах, иначе при произнесении синонима поля будет непонятно, к какому именно полю относится значение.

Для решения этой проблемы введены понятия

- группа элементов на форме
- фокуса элемента.

Они позволяют чётко определять, элементы какой группы пользователь заполняет в данный момент.

Подробные сведения о формате описания группировки элементов и управлении фокусом приведены в описании класса `ContainerMetadata`.

Все элементы на форме, описанные в метаданных, можно представить в виде дерева. Корневым узлом является сама форма, обычными узлами являются группы, а листьями — поля на форме.

Узлы — это элементы, содержащие другие дочерние элементы.
Листья — это элементы, не содержащие дочерних элементов.

Ниже рассмотрен алгоритм обхода дерева для поиска элементов в описании формы, соответствующих фразе пользователя.

Кратко основную идею можно объяснить правилом: «При заполнении полей, находящихся в какой-либо группе, следует произнести название этой группы. Тогда она станет активной».

После распознания фразы пользователя возникает несколько **гипотез** заполнения полей (в случае если произнесенная фраза соответствует сразу нескольким полям).

Гипотеза — это список групп, полей и их значений, которым *может* соответствовать произнесенная пользователем фраза.

Чтобы определить, какое поле заполнять, осуществляется поиск по дереву описания формы (метаданным). Обход дерева начинается с **активного элемента**. В самом начале этим **активным элементом** будет сама форма. Далее у **активного элемента** ищется родительская группа. Группой может быть и сам элемент, если он не является листом дерева.

После этого для каждой из гипотез выбирается **первый элемент гипотезы** и в рамках него ищется соответствующий гипотезе узел или лист (при этом дочерние узлы не обходятся).

Если для **очередного элемента** какой-либо гипотезы найден соответствующий узел или лист в данной группе, то все гипотезы, для которых не был найден соответствующий узел или лист, отбрасываются.

Если в рамках **активного узла** ни для одной из гипотез очередной элемент не найден, то осуществляется переход к родительскому узлу текущего активного узла и поиск повторяется уже в нём.

Поиск продолжается до тех пор, пока всем элементам из оставшихся гипотез не будет найдено соответствие, либо пока не будет достигнут корневой узел (при этом все гипотезы отбрасываются).

Если в конце поиска остался единственный вариант заполнения полей, то эти поля и заполняются.

Если таких гипотез окажется несколько, то приходит уведомление о распознавании нескольких значений.

Если гипотез нет вообще, то фраза пользователя игнорируется.

Класс ValuedFormElementMetadata

Класс описывает типы заполняемых полей на форме. Набор полей различается для различных типов, однако все они должны содержать поле `Type`. Наследник `ContainerMetadata`			
Поле	**Тип поля**	**Обязательное**	**Описание**
Type	`Int32`	Да, за исключением сокращенного формата записи	Тип поля
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю. Например, во фразе «продольный размер 35 миллиметров» слово «миллиметров» необязательно к произношению и не влияет на итоговое значение поля (35). Но, скорее всего, пользователь произнесёт это слово. Поле не используется для строкового типа
Description	`String`	Нет	Описание поля, которое видно по команде "Подсказка"
AvailableValues	Список элементов состоящий из `KeyValuePair <string, string>`	Только для Type=7	Список возможных значений в списке
HandbookKey	`String`	Только для Type=7	Ключ-идентификатор справочника – уникальное значение, однозначно идентифицирующий этот справочник. Значение должно содержать только символы латинского алфавита, нижнее подчеркивание и цифры. UUID и GUID использовать нельзя из-за символа «-»
MultiSelect	`Boolean`	Нет, но применимо только для Type=7	Признак, что в списке можно выбрать несколько значений одновременно. По умолчанию имеет значение `false`
Repeat	`Int32`	Нет, но применимо только для Type= 2 и Type= 3	Количество повторений значений, если поле предназначено для массива вещественных и целых чисел.
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

Text

Простое текстовое поле описывается классом, унаследованным от `ValuedFormElementMetadata`.

При этом существует две равноправные формы описания:

- полная форма, содержащая `key`, `synonyms`, `type` и, возможно, `postfix`,
- сокращённая форма, содержащая лишь `key` и `value`.
- Сокращённая форма описания текстовых полей предоставляет возможность не указывать поля `Type`, `Synonyms` и `Postfix`. В этом случае единственный синоним задается значением поля `Value`, а текстовый тип данных подразумевается.

Полная форма

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор поля в ИС
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Type	`Int32`	Да	Тип поля установлен 0
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

```
{
    "Type": 0,
    "Synonyms": [
        "Описание"
    ],
    "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
}
```

Сокращённая форма

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор типа формы. По этому идентификатору кэшируются метаданные
Value	`String`	Да	Произносимое название поля

```
{
    "Value": "Описание",
    "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<String>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 0,
        "Value": "Надиктованный текст",
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
    }
}
```

Boolean

Логическое поле, которое должно содержать логическое значение `true` или `false`, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор поля в ИС.
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю. Но, скорее всего, пользователь произнесёт это слово
Type	`Int32`	Да	Тип поля установлен `1`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

```
{
    "Type": 1,
    "Synonyms": [
        "Наличие страховки",
        "Страховка",
        "Страховочный ремень"
    ],
    "Key": "1d77096e-53eb-48ea-bfad-4ea3bab2b3a7"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<Boolean>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 1,
        "Value": true,
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
    }
}
```

Int32

Поле с целочисленным значением, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор поля в ИС.
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю. Например, во фразе «Высота 182 миллиметров» слово «миллиметров» необязательно к произношению и не влияет на итоговое значение поля (182). Но, скорее всего, пользователь произнесёт это слово
Type	`Int32`	Да	Тип поля установлен `2`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`
Repeat	`Int32`	Нет	Количество повторений значений, если поле предназначено для массива целых чисел.

```
{
    "Type": 2,
    "Synonyms": [
        "Высота"
    ],
    "Postfix": [
        "миллимтер",
        "миллимтера",
        "миллимтеров"
    ],
    "Key": "b5dcd5b9-576a-420b-aea2-b81171a2ee8e"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<Int32>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 2,
        "Value": 500,
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
    }
}
```

Decimal

Поле с числами в десятичном формате, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор поля в ИС
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля.

			Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю. Но, скорее всего, пользователь произнесёт это слово
Type	`Int32`	Да	Тип поля установлен `3`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`
Repeat	`Int32`	Нет	Количество повторений значений, если поле предназначено для массива вещественных чисел.

```
{
    "Type": 3,
    "Synonyms": [
        "Концентрация раствора"
    ],
    "Postfix": [
        "процент",
        "процента",
        "процентов"
    ],
    "Key": "93d60cc9-fa05-4cdc-b9dd-628e95fe51d9"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<Decimal>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 3,
        "Value": 37.6,
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
    }
}
```

Date

Поле, содержащее дату, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор типа формы
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля/группы. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю. Например, во фразе «Дата осмотра 29 января 2023 года» слово «года» необязательно к произношению и не влияет на итоговое значение поля (29.01.2023). Но, скорее всего, пользователь произнесёт это слово
DateTimeFormat	`String`	Да	Формат, в котором должен возвращаться результат
Type	`Int32`	Да	Тип поля установлен `4`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

```
{
    "DateTimeFormat": "dd.MM.yyyy",
    "Type": 4,
    "Synonyms": [
        "Дата осмотра",
        "Осморт произведён"
    ],
    "Postfix": [
        "год",
        "года"
    ],
    "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<String>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 4,
        "Value": "20.12.2018",
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
    }
}
```

Time

Поле, содержащее время, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор типа формы
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля/группы. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания
DateTimeFormat	`String`	Да	Формат, в котором должен возвращаться результат
Type	`Int32`	Да	Тип поля установлен `5`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

```
{
    "DateTimeFormat": "hh:mm",
    "Type": 5,
    "Synonyms": [
        "Время осмотра",
        "Время",
        "Текущее время"
    ],
    "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<String>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 5,
        "Value": "17:45",
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
    }
}
```

DateTime

Поле, содержащее дату и время, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание

Key	`String`	Да	Идентификатор типа формы. По этому идентификатору кэшируются метаданные
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля/группы. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю
DateTimeFormat	`String`	Да	Формат, в котором должен возвращаться результат
Type	`Int32`	Да	Тип поля установлен `6`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

```
{
    "DateTimeFormat": "dd.MM.yyyy hh:mm",
    "Type": 6,
    "Synonyms": [
        "Дата осмотра",
        "Осморт произведён"
    ],
    "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<String>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 6,
        "Value": "20.12.2018 17:45",
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
    }
}
```

Link

Поле с типом перечисление или ссылка на значения справочника, унаследованное от `ValuedFormElementMetadata`.

Заранее импортированные справочники могут использоваться в различных формах по их идентификатору `HandbookKey`. При этом уникальный идентификатор справочника, определённого в описании формы, должен быть уникальным и не должен совпадать с идентификаторами других справочников.

Справочники,

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор поля в ИС
Type	`Int32`	Да	Тип поля установлен `7`
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю
Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания
HandbookKey	`String`	Да	Ключ-идентификатор справочника – уникальное значение, однозначно идентифицирующий этот справочник. Значение должно содержать только символы латинского алфавита, нижнее подчеркивание и цифры. UUID и GUID использовать нельзя из-за символа «-»
AvailableValues	Список элементов состоящий из `KeyValuePair <string, string>`	Да	Список возможных значений
FreeTextAvaliable	`Boolean`	Нет	Разрешение ввода произвольного текста. По умолчанию имеет значение `false`
MultiSelect	`Boolean`	Нет	Признак, что в списке можно выбрать несколько значений одновременно. По умолчанию имеет значение `false`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

Обратите внимание, что поле `FreeTextAvaliable` по умолчанию имеет значение `false`. В исключительных случаях ему можно задать значение `true`. Это позволит заполнить данное поле не только одним из предустановленных вариантов, но и произвольным текстом (как будто это строковое поле).
Однако такой «свободный» ввод требует от пользователя особого «раздельного» произнесения: он должен произнести название поля, затем сделать паузу, убедиться, что поле стало активным, и лишь затем произнести произвольный текст.

```
{
    "HandbookKey": "38d61a2b_cde8_4d17_b806_60c5fa8c8769",
    "AvailableValues": [
        {
            "Key": "0",
            "Value": "Незначительная",
            "Synonyms": [
                "Слабовязкая"
            ]
        },
        {
            "Key": "1",
            "Value": "Малая",
            "Synonyms": [
                "Маловязкая"
            ]
        },
        {
            "Key": "2",
            "Value": "Повышенная",
            "Synonyms": [
                "Вязкая"
            ]
        },
        {
            "Key": "3",
            "Value": "Высокая",
            "Synonyms": [
                "Высоковязкая"
            ]
        }
    ],
    "Type": 7,
    "Synonyms": [
        "Величина вязкости"
    ],
    "FreeTextAvaliable": false,
    "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<KeyValuePair<string, string>[]>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 7,
        "HandbookRow": [
            {
                "Key": "0",
                "Value": "Незначительная"
            }
        ],
    "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
  }
}
```

При распознавании в поле свободного текста:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 7,
        "Value": "Какой-то произвольный текст",
        "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
    }
}
```

TimeSpan

Поле, содержащее временной интервал, унаследованное от `ValuedFormElementMetadata`.

Поле	Тип поля	Обязательное	Описание
Key	`String`	Да	Идентификатор типа формы. По этому идентификатору кэшируются метаданные
Synonyms	Список строк	Да	Набор всех возможных вариантов произнесения одного и того же поля/группы. Коллекция строк, по которым система будет определять, что распознаваемые данные соответствуют данному элементу или полю

Postfix	Список строк	Нет	Коллекция строк, которые могут быть произнесены после значения, но не должны повлиять на результат распознавания. Это могут быть единицы измерения, которые будет привычно произносить пользователю
Type	`Int32`	Да	Тип поля установлен `8`
IgnoreInAutoFocus	`Boolean`	Нет	Признак, что при последовательном заполнении поле можно пропустить. По умолчанию имеет значение `false`

```
{
    "Type": 8,
    "Synonyms": [
        "Возраст породы"
    ],
    "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
}
```

Ответ от Voice2X

При распознавании возвращается в виде ValuedElementInfo<String>:

```
{
    "MessageType": 1,
    "Data": {
        "Type": 8,
        "Value": "45 лет",
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
    }
}
```

Класс FormSettings

Параметры позволяют указать дополнительные свойства заполнения формы: управлять TTS (голосовым подтверждением ввода) и задавать правила смены фокуса элементов.

Если параметры не были переданы, то будут использоваться значения по умолчанию.

Класс содержит дополнительные параметры заполнения формы			
Поле	Тип поля	Значение по умолчанию	Описание
TtsFieldNameEnabled	`Boolean`	`false`	Озвучивать название поля (первый синоним)
TtsFullGroupNamingEnabled	`Boolean`	`false`	Добавить к названию поля название группы. Используется только если TtsFieldNameEnabled == true
TtsFieldValueEnabled	`Boolean`	`false`	Озвучивать распознанное значение поля (кроме текстовых полей)
TtsTextFieldValueEnabled	`Boolean`	`false`	Озвучивать распознанное значение текстового поля. Используется только если TtsFieldValueEnabled == true
TtsCustomCommandsEnabled	`Boolean`	`false`	Озвучивать распознанные команды
EnableVoiceNavigation	`Boolean`	`true`	Переводить фокус на поле при произнесении его названия
AutoSwitchingEnable	`Boolean`	`false`	Автоматически переводить фокус на следующее поле при заполнении значения поля (кроме текстовых полей и полей с множественными значениями)
AutoSwitchingEnableForText	`Boolean`	`false`	Автоматически переводить фокус на следующее поле при заполнении значения текстового поля
AutoSwitchingEnableForMultiselect	`Boolean`	`false`	Автоматически переводить фокус на следующее поле при заполнении значения поля с множественными значениями
AutoStopOnLastElementFill	`Boolean`	`true`	Останавливать распознавание после заполнения последнего элемента формы, когда включен AutoSwitchingEnable true – останавливать распознавание

			false — не останавливать распознавание, фокус переходит в null, клиенту приходит сообщение

```
"FormSettings": {
   "TtsFieldNameEnabled": true,
   "TtsFieldValueEnabled": true,
   "TtsTextFieldValueEnabled": true,
   "EnableVoiceNavigation": true,
   "AutoSwitchingEnable": true,
   "AutoStopOnLastElementFill": true
},
```

Класс FormCommandSetMetadata

Форма может использовать несколько наборов команд с параметрами для управления заполнения формы, переключаться между которыми можно в процессе заполнения. Параметры произносятся после команды: например, «Перейти к странице пять». Здесь «Перейти к странице» — команда, «пять» — параметр. Голосовая команда может не иметь параметров.

Собственные голосовые команды более приоритетны, чем поля формы. Если название поля случайно совпадёт с названием собственной голосовой команды, то будет возвращена именно голосовая команда.

Набор команд содержит список элементов типа `FormElementMetadata` со следующими ограничениями:

- составное поле типа `Complex` может иметь только один уровень вложенности;
- нельзя использовать группы.

При распознавании фразы, соответствующей какой-либо команде, Voice2X отправляет:

- уведомление о произнесении команды;
- уведомление о произнесении команды (без параметров).

Отдельно импортированный набор команд может использоваться с различными формами по его уникальному ключу. При этом набор команд, заданный в рамках описании формы, должен иметь уникальный ключ и не должен совпадать с другими ключами.

Если указан только `Key`, а поле `Childs` отсутствует, то Voice2X использует импортированный ранее набор команд.

Класс содержит описание набора команд формы			
Поле	**Тип поля**	**Обязательное**	**Описание**
Key	`String`	Нет	Идентификатор набора команд. Он используется для предварительного импорта набора команд и его последующего использования
Childs	Список из FormElementMetadata	Нет	Коллекция команд в наборе. Если не указан, то используется предварительно импортированный список команд. Тип (`Type`) указывает, какие данные должны возвращаться в качестве параметра при произнесении команды с параметром
CultureCode	`String`	Нет	Код культуры, в соответствии с которой будут сериализованы распознанные данные. Например: “ru-RU” Должен совпадать с CultureCode формы

```
{
   "FormCommands":[
      {
         "Key":"Set1",
         "Childs":[
            {
               "Type":0,
               "Key":"Finish",
               "Synonyms":[
                  "Завершить"
               ]
            },
            {
               "Type":2,
               "Key":"GoTo",
               "Synonyms":[
                  "Перейти к странице"
               ]
            }
         ]
      },
      {
         "Key":"Set2",
         "Childs":[
            {
               "Type":0,
               "Key":"Decline",
               "Synonyms":[
                  "отклонить"
               ]
            },
            {
               "Type":0,
               "Key":"Upload",
               "Synonyms":[
                  "загрузить"
               ]
            }
         ]
      }
   ]
}
```

Краткий пример формы

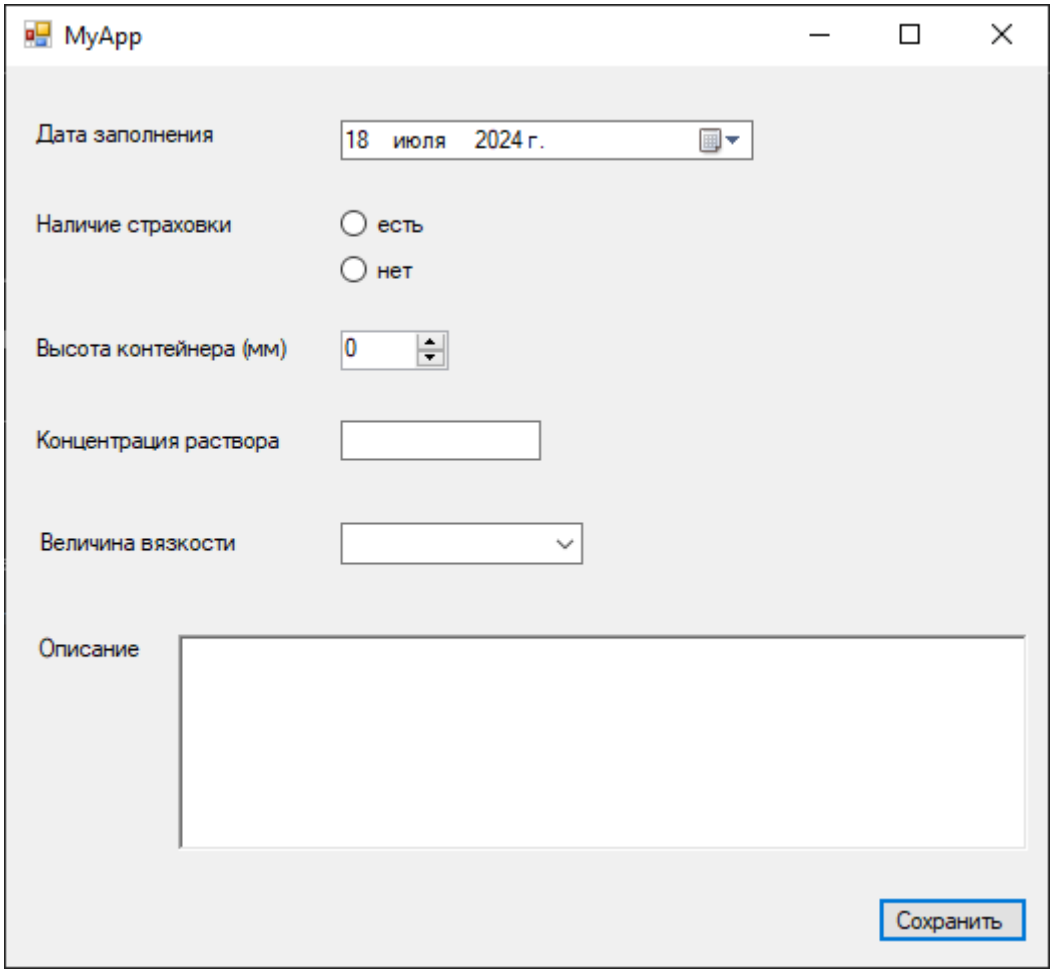


Рисунок 9 – Пример формы

Если в информационной системе форма имеет вид, представленный на рисунке, то JSON-описание формы, которое ИС будет передать в Voice2X, в поле `Data`, должно иметь следующий вид:

```
{
   "Key": "a0bc1744-010c-4414-84e6-acabeff7b517",
   "Name": "Форма MyApp",
   "Childs": [
      {
        "DateTimeFormat": "dd.MM.yyyy",
        "Type": 4,
        "Synonyms": [
          "Дата заполнения"
        ],
        "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542"
      },
      {
        "Type": 1,
        "Synonyms": [
          "Наличие страховки",
          "Страховка"
        ],
        "Key": "1d77096e-53eb-48ea-bfad-4ea3bab2b3a7"
      },
      {
        "Type": 2,
        "Synonyms": [
          "Высота контейнера",
          "Высота"
        ],
        "Key": "b5dcd5b9-576a-420b-aea2-b81171a2ee8e"
      },
      {
        "Type": 3,
        "Synonyms": [
          "Концентрация раствора",
          "Высота"
        ],
        "Key": "93d60cc9-fa05-4cdc-b9dd-628e95fe51d9"
      },
      {
        "HandbookKey": "38d61a2b_cde8_4d17_b806_60c5fa8c8769",
        "AvailableValues": [
        {
            "Key": "0",
            "Value": "Незначительная",
            "Synonyms": [
                "Слабовязкая"
            ]
        },
        {
            "Key": "1",
            "Value": "Малая",
            "Synonyms": [
                "Маловязкая"
            ]
        },
        {
            "Key": "2",
            "Value": "Повышенная",
            "Synonyms": [
                "Вязкая"
            ]
        },
        {
            "Key": "3",
            "Value": "Высокая",
            "Synonyms": [
                "Высоковязкая"
            ]
        }
          ],
        "Type": 7,
        "Synonyms": [
          "Величина вязкости"
        ],
        "Key": "7b43b0e0-5e02-4422-984a-e4f531021744"
      },
      {
        "Type": 0,
        "Synonyms": [
          "Описание"
        ],
        "Key": "9cb3cf80-8f5d-4bdd-9956-16aecb405f3a"
      }
   ],
   "CultureCode": "ru-RU"
}
```

Список REST-методов

Для некоторых REST-запросов предусмотрена альтернатива в виде WebSocket-запросов.
Документ может содержать не полный список REST-запросов.

Протокол взаимодействия: HTTPS.

Формат сообщений: JSON, кодировка UTF-8. Названия полей в JSON-структурах должны рассматриваться как регистронезависимые.

Вызов функций производится посредством запроса HTTP по следующему шаблону:

`<метод><протокол>://<хост>:<порт>/api/v<версия>/<запрос>`

Здесь:

<метод> — используемый метод HTTP (POST, GET, PATCH) зависит от вызываемой функции;
<протокол> — *https*;
<хост> — *доменное имя, иногда IP-адрес ресурса, зависит от службы, к которой выполняется обращение*;
<порт> — порт REST-контроллера службы;
<версия> — версия реализации функции, приводится в описании;
<запрос> — зависит от вызываемой функции и приводится в описании.

Пара **<хост>:<порт>** каждой службы здесь и далее обозначена следующим образом:

- *{AuthenticationServiceAddresses}* — хост и порт службы аутентификации;
- *{BalancerAddresses}* — хост и порт службы балансировки нагрузки;
- *{DBServiceAddresses}* — хост и порт службы взаимодействия с реляционной БД;
- *{EDBServiceAddresses}* — хост и порт службы взаимодействия с нереляционной БД;
- *{FeedbackServiceAddresses}* — хост и порт службы отправки обратной связи;
- *{ReportServiceAddresses}* — хост и порт службы формирования отчетов;
- *{StatisticServiceAddresses}* — хост и порт службы статистики;
- *{DSRESTAddress}* — хост и порт службы бизнес-логики распознавания речи;
- *{TaskServiceAddresses}* — хост и порт службы управления задачами.

Примеры вызова:

`https://{AuthenticationServiceAddresses}/api/v1/Account/Login`

Описание методов сгруппировано по службам, к которым они относятся.

Методы DBService

Группа методов, предоставляемых службой взаимодействия с реляционной БД.

Получение конфигурации

GET {DBServiceAddresses}/api/v1/Settings/ConnectionAddresses

Назначение

Получение конфигурации системы.

Заголовок запроса

Авторизация не требуется.

Поля запроса

Нет.

Пример запроса:

```
curl --location 'https://srvdbs.speechpro.com:39444/api/v1/Settings/ConnectionAddresses'
```

Поля ответа

В структуре `Data` возвращается набор URI для каждого компонента Voice2X:

- `AuthenticationServiceAddresses` — служба аутентификации;
- `BalancerAddresses` — служба балансировки нагрузки;
- `DBServiceAddresses` — служба взаимодействия с реляционной БД;
- `EDBServiceAddresses` — служба взаимодействия с нереляционной БД;
- `FeedbackServiceAddresses` — служба отправки обратной связи;
- `ReportServiceAddresses` — служба формирования отчётов;
- `StatisticServiceAddresses` — служба статистики;
- `TaskServiceAddresses` — служба управления задачами.

Пример:


```
{
    "Data": {
        "BalancerAddresses": [
            "https://srvlbs.speechpro.com:39255"
        ],
        "FeedbackServiceAddresses": [
            "https://srvfbs.speechpro.com:5000"
        ],
        "StatisticServiceAddresses": [
            "https://srvss.speechpro.com:64995"
        ],
        "AuthenticationServiceAddresses": [
            "https://srvauth.speechpro.com:39300"
        ],
        "DBServiceAddresses": [
            "https://srvdbs.speechpro.com:39444"
        ],
        "EDBServiceAddresses": [
            "https://srvedbs.speechpro.com:39600"
        ],
        "TaskServiceAddresses": [
            "https://srvts.speechpro.com:39500"
        ],
        "ReportServiceAddresses": [
            "https://srvrs.speechpro.com:39650"
        ]
    },
    "Error": null
}
```

Методы LoadBalancerService

Группа методов, предоставляемых службой балансировки нагрузки.

Получение рекомендованной DictationService

POST {BalancerAddresses}/api/v1/servers?clientId=

Назначение

Получение сведений о сервисах бизнес-логики распознавания речи, подключённых к службе балансировки нагрузки, и их свойствах.

Параметры запроса

Параметр	Тип	Примечание
clientId	`String`	Уникальный идентификатор пользователя, который должен совпадать с UserId, полученным при аутентификации. Если параметр не передан, то ответ не содержит рекомендованный для подключения сервер (блок `RecommendedServer`), а содержит лишь полный список серверов (блок `ServerList`).

Пример:

```
curl --location --request POST 'https://srvlbs.speechpro.com:39255/api/v1/servers/?
clientId=4e753920-4bf0-4630-9371-79531b1eaf76 \
--header 'Authorization: ••••••'
```

Поля ответа

Поле	Тип	Примечание
TimeStamp	`DateTime`	Текущее время на балансировщике нагрузки
RecommendedServer	`ServerInfo`	Рекомендуемый к использованию сервер
ServerList	Список `ServerInfo`	Список серверов, подключённых к балансировщику нагрузки

Класс ServerInfo имеет следующие поля:

Поле	Тип	Примечание
ServerId	`String`	Уникальный идентификатор сервера
ConnectionString	`String`	Строка для подключения к серверу формата `<адрес сервера>;<WebSocket порт>;<Rest порт>`
CanAcceptClient	`Boolean`	Может ли сервер принять ещё одного клиента
EstimatedLoadLevel	`Decimal`	Предполагаемая загруженность сервера с учётом клиентов, недавно запросивших список серверов
DeclaredLoadLevel	`Decimal`	Заявленная загруженность сервера активными клиентами
LastUpdateDate	`DateTime`	Время последнего обновления данных о сервере на балансировщике нагрузки
LoadPerClient	`Decimal`	Загруженность сервера одним клиентом

Пример:


```
{
    "Data": {
        "TimeStamp": "2024-07-02T09:05:29.3020837Z",
        "RecommendedServer": {
            "ServerId": "d0ace952-234a-4df5-b6fb-5cabcfe2f8fd",
            "ConnectionString": "srvds1.speechpro.com;34000;39255",
            "CanAcceptClient": true,
            "EstimatedLoadLevel": 0.0,
            "DeclaredLoadLevel": 0.0,
```

```
            "LastUpdateDate": "2024-07-02T09:04:46.0349541Z",
            "LoadPerClient": 0.0625
        },
        "ServerList": [
            {
                "ServerId": "d0ace952-234a-4df5-b6fb-5cabcfe2f8fd",
                "ConnectionString": "srvds1.speechpro.com;34000;39255",
                "CanAcceptClient": true,
                "EstimatedLoadLevel": 0.0,
                "DeclaredLoadLevel": 0.0,
                "LastUpdateDate": "2024-07-02T09:04:46.0349541Z",
                "LoadPerClient": 0.0625
            },
            {
                "ServerId": "5af5b482-87fa-4ea4-acaf-330c2b2fd144",
                "ConnectionString": "srvds2.speechpro.com;34000;39255",
                "CanAcceptClient": true,
                "EstimatedLoadLevel": 0.1,
                "DeclaredLoadLevel": 0.0,
                "LastUpdateDate": "2024-07-02T09:04:00.0494181Z",
                "LoadPerClient": 0.1
            }
        ]
    }
}
```

Методы AuthenticationService

Группа методов, предоставляемых службой аутентификации.

Аутентификация и получение токенов

POST {AuthenticationServiceAddresses}/api/v1/Account/Login

Назначение

Аутентификация пользователя и получение токенов AccessToken и RefreshToken.

Заголовок запроса

Content-Type: application/json. Авторизация не требуется.

Тело запроса

Тело запроса должно содержать json-структуру с данными пользователя.

Поле	Тип	Примечание
Login	String	Имя аккаунта, зарегистрированное для использования внешней информационной системой
Password	String	SHA256 от пароля, зарегистрированного для использования внешней информационной системой, приведённое к верхнему регистру

Пример запроса:

```
curl --location 'https://srvauth.speechpro.com:39300/api/v1/Account/Login' \
--header 'Content-Type: application/json' \
--data '{
    "Login":"user",
    "Password":"0CAC6F785283427E764801EF8F2B5845733FF513F59F88765706F9C5B7C14D67"
}'
```

Поля ответа

В структуре `Data` возвращается набор данных:

- `TokenModel` — структура с токенами;
- `UserId` — уникальный идентификатор пользователя;
- `Permissions` — код набора привилегий пользователя в Voice2X.

Структура с токенами содержит:

- `AccessToken`;
- `RefreshToken`;
- `AccessTokenValidTo` — срок действия токена `AccessToken`. По истечении этого времени использование токена `AccessToken` будет невозможно. Потребуется повторная авторизация или перевыпуск токена без предъявления логина и пароля;
- `RefreshTokenValidTo` — срок действия токена `RefreshToken`. По истечении этого времени использование токена `RefreshToken` будет невозможно. Потребуется повторная авторизация.

Пример:

```
{
    "Data": {
        "TokenModel": {
            "AccessToken": "••••••",
            "RefreshToken": "••••••",
            "AccessTokenValidTo": "2024-07-02T07:24:37.1680643Z",
            "RefreshTokenValidTo": "2024-07-02T19:00:00Z"
```

```
            },
            "UserId": "4e753920-4bf0-4630-9371-79531b1eaf76",
            "Permissions": 18446744073709551615
        }
    }
```

Перевыпуск токенов

POST {AuthenticationServiceAddresses}/api/v1/Account/RefreshSession

Назначение

Перевыпуск токенов без предъявления логина и пароля.

Заголовок запроса

Указание типа запроса — application/json и данные для Bearer-авторизации.

Тело запроса

Тело запроса должно содержать json-структуру с текущими токенами.

Поле	Тип	Примечание
AccessToken	String	Текущий токен AccessToken
RefreshToken	String	Текущий токен RefreshToken

Пример запроса:

```
curl --location 'https://srvauth.speechpro.com:39300/api/v1/Account/RefreshSession' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer ••••••' \
--data '{
    "AccessToken":"••••••",
    "RefreshToken": "••••••"
}'
```

Поля ответа

В структуре `Data` возвращается набор данных:

- `AccessToken`;
- `RefreshToken`;
- `AccessTokenValidTo` — срок действия токена `AccessToken`. По истечении этого времени использование токена будет невозможно. Потребуется повторная авторизация или перевыпуск токена без предъявления логина и пароля;
- `RefreshTokenValidTo` — срок действия токена `RefreshToken`. По истечении этого времени использование токена `RefreshToken` будет невозможно. Потребуется повторная авторизация.

Пример:

```
{
    "Data": {
        "AccessToken": "••••••",
        "RefreshToken": "••••••",
        "AccessTokenValidTo": "2024-07-02T07:56:28.8855624Z",
        "RefreshTokenValidTo": "2024-07-02T19:00:00Z"
    }
}
```

Методы DictationService

Группа методов, предоставляемых службой бизнес-логики распознавания речи.

Справочники

Методы для работы со справочниками.

Получение списка импортированных справочников

GET {DSRESTAddress}/api/Handbook

Назначение

Получение списка импортированных справочников и их значений.

Пример запроса

```
https://SRVDS1:39255/api/Handbook
```

Поля ответа

Массив элементов, основанных на классе `HandbookMetadata`, с переименованным в `Id` полем `HandbookKey`.

Пример:


```
{
  "Data": [
    {
      "Id": "8ecafe5b-a6cf-4576-b423-cd915d25bba3",
      "Name": null,
      "AvailableValues": [
        {
          "Key": "0",
          "Value": "Нормальное"
        },
        {
          "Key": "1",
          "Value": "Повышенное"
        },
        {
          "Key": "2",
          "Value": "Пониженное"
        }
      ]
    },
    {
      "Id": "3b00e940-1f75-4bae-ae33-7796dbceb768",
      "Name": null,
      "AvailableValues": [
        {
          "Key": "0",
          "Value": "Нормальное"
        },
        {
          "Key": "1",
          "Value": "Повышенное"
        },
        {
          "Key": "2",
          "Value": "Пониженное"
        }
      ]
    },
    {
      "Id": "38d61a2b-cde8-4d17-b806-60c5fa8c8769",
      "Name": null,
      "AvailableValues": [
        {
          "Key": "0",
          "Value": "Мужской"
        },
        {
          "Key": "1",
          "Value": "Женский"
        }
      ]
    }
  ],
  "Error": null
}
```

Удаление импортированного справочника

DELETE {DSRESTAddress}/api/Handbook

Назначение

Удаление справочника.

Параметры запроса

Параметр	Тип	Примечание
Id	String	Ключ-идентификатор удаляемого справочника

Пример запроса

```
curl --location --request DELETE 'https://SRVDS1:39255/api/Handbook?Id=38d61a2b-cde8-4d17-b806-60c5fa8c8769'
```

Поля ответа

В `Data` возвращается:

- `true` — если справочник был найден и помечен на удаление;
- `false` — если справочник с таким идентификатором не найден.

Пример:

```
{
    "Data": true,
    "Error": null
}
```

Описание формы

Методы для работы с описанием формы.

Получение списка импортированных форм

GET {DSRESTAddress}/api/FormMetadata

Назначение

Получение списка импортированных форм.

Пример запроса

```
https://SRVDS1:39255/api/FormMetadata
```

Поля ответа

Массив элементов, основанных на классе FormMetadata: полное описание кешированных форм.

Пример:


```
{
    "Data": [
        {
            "Key": "a0bc1744-010c-4414-84e6-acabeff7b517",
            "Name": "Форма №1",
            "Childs": [
                {
                    "FreeTextAvaliable": false,
                    "IgnoreInAutoFocus": false,
                    "Type": 4,
                    "Key": "0d2f0d70-be6a-4f02-b71f-b3e2adc80542",
                    "Synonyms": [
                        "Дата осмотра",
                        "Дата проведения осмотра"
                    ],
                    "Postfix": [],
                    "MultiSelect": false,
                    "DateTimeFormat": "dd.MM.yyyy"
                }
            ]
        }
    ]
}
```


Удаление импортированной формы

DELETE {DSRESTAddress}/api/FormMetadata

Назначение

Удаление импортированной формы.

Параметры запроса

Параметр	Тип	Примечание
Id	String	Ключ-идентификатор удаляемой формы

Пример запроса

```
curl --location --request DELETE 'https://SRVDS1:39255/api/FormMetadata?Id=a0bc1744-010c-4414-84e6-acabeff7b517'
```

Поля ответа

В `Data` возвращается:

- `true` — если форма была найдена и помечена на удаление;
- `false` — если форма с таким идентификатором не найдена.

Пример:

```
{
    "Data": true,
    "Error": null
}
```

Наборы команд

Методы для работы с наборами команд.

Получение списка импортированных наборов команд

GET {DSRESTAddress}/api/FormCommandSet

Назначение

Получение списка импортированных наборов команд.

Пример запроса

```
https://SRVDS1:39255/api/FormCommandSet
```

Поля ответа

Массив элементов, основанных на классе FormCommandSetMetadata: полное описание кешированных наборов команд.

Пример:


```
{
  "Data": [
    {
      "Key": "Set1",
      "Name": null,
      "IsAutogenerated": false,
      "Childs": [
        {
          "Childs": null,
          "Type": 0,
          "Synonyms": [
            "Завершить"
          ],
          "Postfix": [],
          "Key": "Finish",
          "TabIndex": null,
          "HandbookKey": null,
          "Name": null,
          "AvailableValues": null,
          "DateTimeFormat": null,
          "TimeSpanFormat": null
        }
      ]
    },
    {
      "Key": "Set2",
      "Name": null,
      "IsAutogenerated": false,
      "Childs": [
        {
          "Childs": null,
          "Type": 0,
          "Synonyms": [
            "отклонить"
          ],
          "Postfix": [],
          "Key": "Decline",
          "TabIndex": null,
          "HandbookKey": null,
          "Name": null,
          "AvailableValues": null,
          "DateTimeFormat": null,
          "TimeSpanFormat": null
        },
        {
          "Childs": null,
          "Type": 0,
          "Synonyms": [
            "загрузить"
          ],
          "Postfix": [],
          "Key": "Upload",
          "TabIndex": null,
          "HandbookKey": null,
          "Name": null,
```

```
            "AvailableValues": null,
            "DateTimeFormat": null,
            "TimeSpanFormat": null
          }
        ]
      },
      {
        "Key": "Set3",
        "Name": null,
        "IsAutogenerated": false,
        "Childs": [
          {
            "Childs": null,
            "Type": 2,
            "Synonyms": [
              "перейти к странице"
            ],
            "Postfix": [],
            "Key": "GoTo",
            "TabIndex": null,
            "HandbookKey": null,
            "Name": null,
            "AvailableValues": null,
            "DateTimeFormat": null,
            "TimeSpanFormat": null
          },
          {
            "Childs": null,
            "Type": 0,
            "Synonyms": [
              "очистить"
            ],
            "Postfix": [],
            "Key": "Clear",
            "TabIndex": null,
            "HandbookKey": null,
            "Name": null,
            "AvailableValues": null,
            "DateTimeFormat": null,
            "TimeSpanFormat": null
          }
        ]
      },
      {
        "Key": "TemplateApp_standart",
        "Name": null,
        "IsAutogenerated": false,
        "Childs": [
          {
            "Childs": null,
            "Type": 0,
            "Synonyms": [
              "Закончить ввод"
            ],
            "Postfix": [],
            "Key": "Закончить_ввод",
            "TabIndex": null,
            "HandbookKey": null,
            "Name": null,
            "AvailableValues": null,
            "DateTimeFormat": null,
            "TimeSpanFormat": null
          },
          {
            "Childs": null,
            "Type": 0,
            "Synonyms": [
              "Очистить поле"
            ],
            "Postfix": [],
            "Key": "Очистить_поле",
            "TabIndex": null,
            "HandbookKey": null,
            "Name": null,
            "AvailableValues": null,
            "DateTimeFormat": null,
            "TimeSpanFormat": null
          }
        ]
      }
    ],
    "Error": null
  }
```

Удаление импортированного набора команд

DELETE {DSRESTAddress}/api/FormCommandSet

Назначение

Удаление импортированного набора команд.

Параметры запроса

Параметр	Тип	Примечание
Id	String	Ключ-идентификатор удаляемого набора команд

Пример запроса

```
curl --location --request DELETE 'https://SRVDS1:39255/api/FormCommandSet?Id=Set3'
```

Поля ответа

В `Data` возвращается:

- `true` — если набор команд был найден и помечен на удаление;
- `false` — если набор команд с таким идентификатором не найден.

Пример:

```
{
    "Data": true,
    "Error": null
}
```

Типы полей

Методы для работы с типами полей.

Получение списка доступных типов полей

GET {DSRESTAddress}/api/FormElementType

Назначение

Получение списка доступных типов полей, которые могут использоваться на форме.

Поля запроса

Не требуются.

Пример запроса

```
https://SRVDS1:39255/api/FormElementType
```

Поля ответа

Массив описаний полей.

Поле	Тип	Примечание
Id	Int32	Идентификатор типа данных
Name	String	Наименование типа поля
IsDeleted	Boolean	Параметр, который показывает, удалено поле или нет
XmlGrammar	String, Null	Строка с SRGS-грамматикой. В тегах, в переменной `out` возвращается результирующее распознанное значение. Если грамматика отсутствует, возвращается значение Null

Пример:


```
{
  "Data": [
    {
      "Id": 0,
      "Name": "Текст",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 1,
      "Name": "Логическое значение",
      "IsDeleted": false,
      "XmlGrammar": "<?xml version=\"1.0\" encoding=\"utf-8\"?>\r\n<grammar xml:lang=\"ru-
RU\"\r\n\troot=\"field_1\"\r\n\tmode=\"voice\"\r\n\tversion=\"1.0\"\r\n\txmlns=\"http://www.w3.org/2001/06
/grammar\"\r\n\ttag-format=\"semantics/1.0-literals\">\r\n\r\n  <rule id=\"field_1\" scope=\"public\">\r\n
     <one-of>\r\n        <item>да<tag>out=true</tag></item>\r\n
<item>есть<tag>out=true</tag></item>\r\n        <item>нет<tag>out=false</tag></item>\r\n
<item>отсутствуют<tag>out=false</tag></item>\r\n        <item>отсутствует<tag>out=false</tag></item>\r\n
    </one-of>\r\n  </rule>\r\n\r\n</grammar>\r\n"
    },
    {
      "Id": 2,
      "Name": "Целое число",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 3,
      "Name": "Дробное число",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 4,
      "Name": "Дата",
      "IsDeleted": false,
      "XmlGrammar": null
    },
```

```
    {
      "Id": 5,
      "Name": "Время",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 6,
      "Name": "Дата и время",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 7,
      "Name": "Перечисление/Справочник",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 8,
      "Name": "Промежуток времени",
      "IsDeleted": false,
      "XmlGrammar": null
    },
    {
      "Id": 9,
      "Name": "Справочник МКБ",
      "IsDeleted": false,
      "XmlGrammar": null
    }
  ],
  "Error": null
}
```


Создание собственного типа поля

POST {DSRESTAddress}/api/FormElementType

Назначение

Создание собственного типа поля для последующего использования на форме.

Поля запроса

Поле	Тип	Примечание
Id	Int32	Идентификатор типа данных. Значение должно быть больше 1000
Name	String	Наименование типа поля
XmlGrammar	String	Строка с SRGS-грамматикой. В тэгах, в переменной `out` возвращается результирующее распознанное значение

Построение SRGS-грамматики для своего собственного типа является нетривиальной задачей. Однако достаточно опытный специалист вполне может реализовать свою грамматику согласно следующим стандартам:

1. Грамматика SRGS: https://www.w3.org/TR/speech-grammar/
2. Семантическая интерпретация SISR: https://www.w3.org/TR/semantic-interpretation/

Формат тегов семантической интерпретации ограничен форматом "semantics/1.0"

Пример запроса

```
curl --location --request POST 'https://SRVDS1:39255/api/FormElementType' \
--header 'Content-Type: application/json' \
--data-raw '{
    "Id": 10001,
    "Name": "SexType",
    "XmlGrammar": "<?xml version=\"1.0\" encoding=\"utf-8\"?>
    <grammar xml:lang=\"ru-RU\" root=\"Sex\" mode=\"voice\" version=\"1.0\"
xmlns=\"http://www.w3.org/2001/06/grammar\" tag-format=\"semantics/1.0\">
```

```
        <rule id=\"Sex\" scope=\"public\">
            <one-of>
                <item>Мужской<tag>out='\''М.'\''</tag></item>
                <item>Женский<tag>out='\''Ж.'\''</tag></item>
            </one-of>
        </rule>
    </grammar>"
}'
```

Поля ответа

В `Data` возвращается:

- `true` — если новый тип данных был создан;
- `false` — если новый тип создать не удалось.

Пример:

```
{
    "Data": true,
    "Error": null
}
```

Удаление собственного типа поля

DELETE {DSRESTAddress}/api/FormElementType

Назначение

Удаление собственного типа поля.

Параметры запроса

Параметр	Тип	Примечание
Id	String	Ключ-идентификатор удаляемого типа поля. Возможно удалить только собственные поля, созданные запросом FormElementType

Пример запроса

```
curl --location --request DELETE 'https://SRVDS1:39255/api/FormElementType?Id=10001'
```

Поля ответа

В `Data` возвращается:

- `true` — если тип данных был найден и помечен на удаление;
- `false` — если тип данных с таким идентификатором не найден.

Пример:

```
{
    "Data": true,
    "Error": null
}
```

Классы сообщений WS

Раздел содержит описание основных классов сообщений, передаваемых через WebSocket.

DictationApiMessage<T>

Класс-контейнер, в который оборачиваются все сообщения			
Поле	Тип поля	Обязательное	Описание
MessageType	`Int32`	Да	Тип сообщения
Data	зависит от сообщения	Нет	Данные, соответствующие типу сообщения

ActivationResult

Класс информации о результате активации лицензии			
Поле	Тип поля	Обязательное	Описание
ResultType	`Int32`	Да	Результат активации лицензии: **0** – активация ещё не начата или не завершена; **1** – активация завершена успешно; **2** – возникла ошибка при попытке активации; **3** – сервер активации недоступен; **4** – активация прервана пользователем; **5** – активация запрещена.
ErrorText	`String`	Да	Текстовое сообщение (пояснение)

AsrSdkWord

Класс распознанных слов			
Поле	Тип поля	Обязательное	Описание
Begin	`Int32`	Да	Число миллисекунд от начала диктовки
Length	`Int32`	Да	Длительность в миллисекундах
Text	`String`	Да	Распознанное слово
Index	`Int32`	Да	
ProbabilityNorm	`Decimal`	Да	
EndSentence	`Int32`	Да	
DictorNum	`Int32`	Да	
PunctuationText	`String`	Да	

CommandInfo

Класс описания голосовых команд			
Поле	Тип поля	Обязательное	Описание
Id	`String`	Да	Уникальный идентификатор голосовой команды. Он возвращается в ответе
Type	`Int32`	Да	Тип голосовой команды: **0** – зарезервировано (не использовать); **1** – команда голосовой активации распознавания; **2** – команда голосовой деактивации распознавания; **3** – пользовательская команда
Pattern	`String`	Да	Произносимый пользователем текст

ErrorInfo

Класс описания возникшей ошибки			
Поле	Тип поля	Обязательное	Описание
Text	`String`	Да	Текстовое сообщение об ошибке
DebugText	`String`	Нет	Расширенное описание ошибки

HandbookMetadata

Класс описания данных справочника			
Поле	Тип поля	Обязательно	Описание
HandbookKey	`String`	Да	Ключ-индентификатор справочника – уникальное значение, однозначно идентифицирующий этот справочник. Значение должно содержать только символы латинского алфавита, нижнее подчеркивание и цифры. UUID и GUID использовать нельзя из-за символа «-»
Name	`String`	Нет	Наименование
AvailableValues	Список элементов состоящий из `KeyValuePair<string, string>`	Да	Список возможных значений в формате “ключ-значение”

NotificationInfo

Класс описания уведомления о событии			
Поле	Тип поля	Обязательное	Описание
Code	Int32	Да	Код события: **1** – сервис распознавания доступен, восстановлен текущий режим распознавания **2** – сервис распознавания временно недоступен **3** – зарезервировано **4** – зарезервировано

ProtectionData

Класс данных ключа защиты			
Поле	Тип поля	Обязательное	Описание
LicenseType	Int32	Да	Тип лицензии: **1** – триальная; **2** – постоянная; **3** – на число запусков; **4** – по времени
Type	Int32	Да	Тип защиты: **0** – не определено; **1** – нет защиты; **2** – HASP.
LastUpdateTimestamp	DateTime	Да	Время последнего получения данных о ключе защиты
CountOfLicenses	Int32	Да	Общее количество лицензий
CountOfCapturedLicenses	Int32	Да	Количество используемых лицензий
DaysLeft	Int32	Да	Количество оставшихся дней. При значении **0** словарь или приложение не смогут быть использованы. При значении **-1** словарь или приложение не имеют ограничения по времени
LaunchCount	Int32	Да	Количество оставшихся запусков. При значении **0** словарь или приложение не смогут быть использованы. При значении **-1** словарь или приложение не имеют ограничения по количеству запусков

ProtectionInfo

Класс информации о лицензиях			
Поле	Тип поля	Обязательное	Описание
Type	Int32	Да	Тип ключа защиты: **0** – лицензия на использование; **1** – лицензия на обновление (в данный момент не используется)
ComponentId	Int32	Да	Идентификатор компонента: **пустая строка** – серверная лицензия; **непустая строка** – идентификатор словаря
ProtectionData	ProtectionData	Да	Данные ключа защиты

RecognitionModeInfo

Класс описания текущего режима распознавания			
Поле	Тип поля	Обязательное	Описание
Mode	Int32	Да	Режим распознавания: **0** – распознавание отключено **1** – голосовая активация **2** – слитное распознавание **3** – свободное заполнение формы **4** – последовательное заполнение формы
Info	String	Да	Идентификатор распознаваемой формы при режиме распознавания 3 и 4
State	Int32	Да	Состояние: **0** – запускается **1** – запущен и работает **2** – остановлен **3** – останавливается

ServerThemesInfo

Класс информации о специальных языковых моделях			
Поле	Тип поля	Обязательное	Описание
ThemeInfos	массив ThemeInfo	Да	Список установленных специальных языковых моделей на выбранном узле распознавания речи

ThemeInfo

Класс специальной языковой модели распознавания			
Поле	Тип поля	Обязательное	Описание
LicenseState	Int32	Нет	Состояние лицензии: **0** – лицензия активна **1** – лицензия закончилась **2** – лицензия не найдена
CompositionType	Int32	Нет	Тип языковой модели распознавания: **0** – базовая; **1** – специализированная; **2** – полная.
Id	String	Да	Идентификатор языковой модели
ThemeId	String	Да	Краткий идентификатор языковой модели
Name	String	Да	Отображаемое пользователю название языковой модели
Version	String	Да	Версия языковой модели в общепринятом формате major.minor.build
ShopUrl	String	Да	
NewVersionInfoUrl	String	Нет	
Description	String	Да	Описание назначения языковой модели

TextResult

Класс результата распознавания			
Поле	Тип поля	Обязательное	Описание
Text	`String`	Да	Распознанный текст
IsFinal	`Boolean`	Да	Тип результата: **true** – окончательный; **false** – промежуточный
Words	массив из `AsrSdkWord`	Да	Массив распознанных слов
CommandId	`String`	Да	Идентификатор команды, если была распознана голосовая команда или пустая строка
Length	`Int32`	Да	Длительность обработанного звука в миллисекундах

TtsStateInfo

Класс состояния синтезатора речи			
Поле	Тип поля	Обязательное	Описание
VoiceList	`KeyValuePair<string, string>`	Нет	Список голосов
SelectedVoice	`KeyValuePair<string, string>`	Нет	Выбранный голос
Speed	`Decimal`	Да	Скорость воспроизведения. Допустимые значения от 0,5 до 2.
State	`Int32`	Да	Статус: **0** – всё работает нормально; **1** – внутренняя ошибка, возможно некорректно установлен синтезатор; **2** – ошибка проверки лицензии

UserNotification

Класс оповещений			
Поле	Тип поля	Обязательное	Описание
Message	`String`	Да	Текст сообщения
Type	`Int32`	Да	Тип сообщения: **1** – информационное сообщение; **2** – предупреждение; **3** – ошибка
Action	`Int32`	Да	Предполагаемое действие: **0** – действий не требуется; **1** – закрыть подключение с текущим сервером и подключиться к следующему доступному серверу