

Функциональное описание и описание методов API

**«Platform-T». Программный комплекс распределенных вычислений и алгоритмической обработки данных**

Содержание

[1. Введение 5](#_Toc110626532)

[1.1. Функциональные модули 5](#_Toc110626533)

[1.1.1. Модуль агрегации данных (Aggregation Module) 5](#_Toc110626534)

[1.1.2. Модуль алгоритмической обработки (Processing) 5](#_Toc110626535)

[1.1.3. Модуль движка АПЗ (ArchiveSearchQueriesEngineModule) 6](#_Toc110626536)

[1.1.4. Модуль движка ППЗ (SearchQueriesModule) 6](#_Toc110626537)

[1.1.5. Модуль системы учета (AccountingModule) 7](#_Toc110626538)

[1.1.6. Модуль хранения бинарных данных (StorageModule) 7](#_Toc110626539)

[2. Aggregation Module 8](#_Toc110626540)

[2.1. Работа с сущностью 8](#_Toc110626541)

[2.1.1. Получение одной сущности по Id 8](#_Toc110626542)

[2.1.2. Создание одной сущности 8](#_Toc110626543)

[2.1.3. Обновление одной сущности 9](#_Toc110626544)

[2.2. Работа с метаинформацией в сущности 10](#_Toc110626545)

[2.2.1. Добавление метаинформации в сущность 10](#_Toc110626546)

[2.2.2. Изменение метаинформации в сущности 10](#_Toc110626547)

[2.3. Операции с метаинформацией в нескольких сущностях 11](#_Toc110626548)

[2.4. Другие методы 12](#_Toc110626549)

[2.4.1. Проверка использования группы доступа в сущностях 12](#_Toc110626550)

[3. Processing 13](#_Toc110626551)

[3.1. IAeDetectorService 13](#_Toc110626552)

[3.2. IFullTextIndexingService 14](#_Toc110626553)

[3.3. IIdentService 15](#_Toc110626554)

[3.4. ITranscriberService 17](#_Toc110626555)

[4. ArchiveSearchQueriesEngineModule 19](#_Toc110626556)

[4.1. IASQRegistry API 19](#_Toc110626557)

[4.1.1. RegisterSearchQueryAsync 19](#_Toc110626558)

[4.1.2. CancelSearchQueryAsync 19](#_Toc110626559)

[4.2. REST API 20](#_Toc110626560)

[4.2.1. Тип 1 20](#_Toc110626561)

[4.2.2. Тип 2 21](#_Toc110626562)

[4.2.3. Тип 3 21](#_Toc110626563)

[5. SearchQueriesModule 22](#_Toc110626564)

[5.1. API SearchQueriesController.ChangeNameOfSearchQueryAsync 22](#_Toc110626565)

[5.1.1. Общая информация 22](#_Toc110626566)

[5.1.2. Параметры 22](#_Toc110626567)

[5.1.3. Возвращаемое значение 29](#_Toc110626568)

[5.1.4. Возвращаемые ошибки 29](#_Toc110626569)

[5.2. API SearchQueriesController.DeleteBatchSearchQueryAsync 30](#_Toc110626570)

[5.2.1. Общая информация 30](#_Toc110626571)

[5.2.2. Параметры 30](#_Toc110626572)

[5.2.3. Возвращаемое значение 31](#_Toc110626573)

[5.2.4. Возвращаемые ошибки 31](#_Toc110626574)

[5.3. API SearchQueriesController.DeleteSearchQueryAsync 32](#_Toc110626575)

[5.3.1. Общая информация 32](#_Toc110626576)

[5.3.2. Параметры 32](#_Toc110626577)

[5.3.3. Возвращаемое значение 33](#_Toc110626578)

[5.3.4. Возвращаемые ошибки 33](#_Toc110626579)

[5.4. API SearchQueriesController.GetMaxCountBySqTypeAsync 34](#_Toc110626580)

[5.4.1. Общая информация 34](#_Toc110626581)

[5.4.2. Параметры 34](#_Toc110626582)

[5.4.3. Возвращаемое значение 35](#_Toc110626583)

[5.4.4. Возвращаемые ошибки 35](#_Toc110626584)

[5.5. API SearchQueriesController.GetSearchQueriesByFilterAsync 36](#_Toc110626585)

[5.5.1. Общая информация 36](#_Toc110626586)

[5.5.2. Параметры 36](#_Toc110626587)

[5.5.3. Возвращаемое значение 42](#_Toc110626588)

[5.5.4. Возвращаемые ошибки 43](#_Toc110626589)

[5.6. API SearchQueriesController.GetSearchQueryByIdAsync 43](#_Toc110626590)

[5.6.1. Общая информация 43](#_Toc110626591)

[5.6.2. Параметры 43](#_Toc110626592)

[5.6.3. Возвращаемое значение 44](#_Toc110626593)

[5.6.4. Возвращаемые ошибки 45](#_Toc110626594)

[5.7. API SearchQueriesController.TryMarkAsClearedAsync 45](#_Toc110626595)

[5.8. Общая информация 45](#_Toc110626596)

[5.8.1. Параметры 45](#_Toc110626597)

[5.8.2. Возвращаемое значение 46](#_Toc110626598)

[5.8.3. Возвращаемые ошибки 46](#_Toc110626599)

[6. AccountingModule 48](#_Toc110626600)

[6.1. Голосовая информация 48](#_Toc110626601)

[6.1.1. IVoiceInfoData 48](#_Toc110626602)

[6.1.2. IVoiceModelData 48](#_Toc110626603)

[6.1.3. IVoiceModelMetaData 49](#_Toc110626604)

[6.1.4. VoiceModelState 50](#_Toc110626605)

[6.1.5. IAudioFileWithSegmentationData 50](#_Toc110626606)

[6.1.6. IAudioMetaData 51](#_Toc110626607)

[6.1.7. IFileMetaData 52](#_Toc110626608)

[6.1.8. ISegmentationData 53](#_Toc110626609)

[6.1.9. ISegmentationMetaData 54](#_Toc110626610)

[6.1.10. ChannelNumber 57](#_Toc110626611)

[6.1.11. ChannelType 58](#_Toc110626612)

[6.1.12. SegmentationAlgorithmType 58](#_Toc110626613)

[6.2. Лицевая информация 58](#_Toc110626614)

[6.2.1. Общая информация 58](#_Toc110626615)

[6.2.2. IFaceInfoData 59](#_Toc110626616)

[6.2.3. IFaceModelData 60](#_Toc110626617)

[6.2.4. IFaceModelMetaData 60](#_Toc110626618)

[6.2.5. IFaceEngineMetaData 61](#_Toc110626619)

[6.2.6. IVideoFileWithSegmentationData 61](#_Toc110626620)

[6.2.7. IFileMetaData 62](#_Toc110626621)

[6.2.8. IVideoMetaData 63](#_Toc110626622)

[6.2.9. IFaceSegmentationData 63](#_Toc110626623)

[6.2.10. IFaceSegmentationMetaData 64](#_Toc110626624)

[6.2.11. IEstimatedDeviation 65](#_Toc110626625)

[6.2.12. IPortraitCharacteristics 66](#_Toc110626626)

[7. StorageModule 68](#_Toc110626627)

[7.1. Тип 1 68](#_Toc110626628)

[7.2. Тип 2 68](#_Toc110626629)

[7.3. Тип 3 69](#_Toc110626630)

[7.4. Тип 4 69](#_Toc110626631)

[7.5. Тип 5 70](#_Toc110626632)

[7.6. Тип 6 70](#_Toc110626633)

[7.7. Тип 7 71](#_Toc110626634)

[7.8. Тип 8 72](#_Toc110626635)

[7.9. Тип 9 72](#_Toc110626636)

[7.10. Тип 10 73](#_Toc110626637)

[7.11. Тип 11 74](#_Toc110626638)

[7.12. Тип 12 74](#_Toc110626639)

[7.13. Тип 13 74](#_Toc110626640)

[7.14. Тип 14 74](#_Toc110626641)

1. Введение

Программный комплекс распределенных вычислений алгоритмической обработки данных «Platform-T» предназначен для создания систем поиска, систем управления загрузкой персонала, систем аналитики, цифровых ассистентов, систем автоматизации документооборота. Функционально комплекс можно разделить на следующие модули:

* Aggregation Module (модуль агрегации данных);
* [Processing (модуль алгоритмической обработки)](https://confluence.speechpro.com/pages/viewpage.action?pageId=187176066);
* [Модуль движка АПЗ (ArchiveSearchQueriesEngineModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178170305);
* [Модуль движка ППЗ (SearchQueriesModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178170302);
* [Модуль системы учета (AccountingModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178168049);
* [Модуль хранения бинарных данных (StorageModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178168051).
  1. Функциональные модули

### Модуль агрегации данных (Aggregation Module)

*Aggregation* – это общий (главный) модуль, выполняющий функцию как агрегации данных, так и распределения вычислительных ресурсов.

Основные функции модуля:

* Фильтрация объектов по коллекциям с целью создания кластеров для распределения вычислительных ресурсов;
* Создание новых пользователей с указанием их прав в рамках комплекса;
* Предоставление функционала по работе с сессией пользователя;
* Предоставление функционала по проверке привилегий и управления приоритетами вычислений;
* Возможность ограничения доступа пользователя к определенному функционалу;
* Группы пользователей;
* Логическое объединение пользователей в группы;
* Механизм назначения ролей и привилегий пользователям с последующим мониторингом нагрузки на комплекс.

### М[одуль алгоритмической обработки](https://confluence.speechpro.com/pages/viewpage.action?pageId=187176066) (Processing)

Processing – модуль, предназначенный для извлечения информации из различных медиаданных с помощью различных алгоритмов с последующим обобщением.

Основные функции модуля:

* Расширение сущностей из Aggregation результатами распознавания речи с последующим разделение голосового коктейля на дикторов (перевод речи в текст происходит на различных языках);
* Просмотр и индексирование автоматически построенных диалогов;
* Расширение сущностей из Aggregation обнаруженными акустическими событиями (детекция речи, определение языка диктора);
* Расширение сущностей из Aggregation биометрическими моделями (голос, лицо и т. д.);
* Просмотр детектированных дикторов, найденных языков, пола дикторов и распознавание образа диктора;
* Распознавание лица и автоматическое создание карточек физических лиц с привязкой к номеру телефона;
* Кэширование голосовых моделей по сущностям из Aggregation и последующая кластеризация;
* Подготовка для поиска в сравнениях по голосу;
* Автоматическое наполнение учетной базы физических лиц;
* Автоматический поиск дубликатов физических лиц на основе сравнения по голосу;
* Чистка и актуализация учетной базы физических лиц.

### [Модуль движка АПЗ (ArchiveSearchQueriesEngineModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178170305)

Модуль АПЗ предназначен для создания/изменения/выполнения/просмотра результатов поисковых запросов по архивным данным.

Основной функционал модуля:

* Механизм агрегированного поиска по Архиву записей одновременно по метаинформации, по документам, по сравнению голосов;
* Возможность задать сложный комбинированный фильтр;
* Возможность хранить результаты для поиска по архиву и делать подзапросы к этим результатам;
* Возможность работать с результатами АПЗ – фильтровать их, проставлять метки, отправлять уведомления о кластеризации;
* Возможность сравнивать новые сущности из Aggregation по сложным критериям – как по метаинформации, так и по документам, а также через сравнение по голосу;
* Хранение и доступ к сообщениям для конкретных пользователей;
* Индексация текстовых документов, полнотекстовый поиск;
* Мини-система почты.

### [Модуль движка ППЗ (SearchQueriesModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178170302)

Модуль ППЗ предназначен для просмотра результатов поисковых запросов по потоковым данным, в том числе в режиме реального времени.

Основной функционал модуля:

* Возможность агрегированного поиска одновременно по метаинформации, по документам, по сравнению голоса и лица;
* Возможность «горячих» проверок новой информации с возможностью простановки меток, отправки уведомлений.

### [Модуль системы учета (AccountingModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178168049)

AccountingModule – модуль, позволяющий создавать различные объекты учета путем группировки различной информации по задаваемым пользователем признакам.

Основной функционал модуля:

* Хранение истории изменения типов объектов;
* Возможность отслеживания изменений по важным для пользователя типам объектов;
* Фильтрация по объектам, структура которых задается через описание;
* Кросс-импорты – передача объектов между различными системами;
* Возможность хранения голосовой, лицевой и прочей информации и поиска по «схожести» этой информации.

### [Модуль хранения бинарных данных (StorageModule)](https://confluence.speechpro.com/pages/viewpage.action?pageId=178168051)

Модуль хранения данных – интеллектуальное файловое хранилище, предназначенное для хранения больших объемов данных.

Основной функционал модуля:

* Загрузка и выгрузка бинарных данных в файловое хранилище;
* Хранение и доступ к бинарным данным;
* Абстракция над файловым хранилищем;
* Возможность работать с различными физическими файловыми хранилищами;
* Возможность «схлопывания» бинарно одинаковых данных на файловом хранилище;
* Уменьшение фактически потребляемого места в файловом хранилище;
* Регистрация пользовательских действий в комплексе;
* Журнал действий пользователя.

1. Aggregation Module
   1. Работа с сущностью

### Получение одной сущности по Id

**Метод:**

Необходимо вызвать метод SaveEntityAsynс:

Task<IResultWithErrorData<ITypedEntity>> GetEntityWithErrorDataAsync(Guid entityId, string sessionIdentifier);

**Параметры:**

* ID - это идентификатор сущности;
* sessionIdentifier - идентификатор сессии клиента.

**Возвращаемыйрезультат:**

* IResultWithErrorData, содержащий экземпляр запрашиваемой ITypedEntity в поле Result в случае успешной операции или информацию об ошибке в поле ErrorData;
* В случае отсутствия сущности возвращается ошибка с кодом 404;
* В случае отсутствия прав доступа пользователя к данной сущности возвращается сущность с некорректными значениями полей, кроме Id и с полем Forbidden = true;
* В случае иных ошибок возвращается ошибка с кодом 500.

### Создание одной сущности

**Метод:**

Необходимо вызвать метод SaveEntityAsynс:

Task<ISuccessData> SaveEntityAsync(ITypedEntity entity, string sessionIdentifier);

**Параметры:**

Необходимо передать сущность (ITypedEntity entity) и id сессии (sessionIdentifier). В entity нужно заполнить поля:

1. Id – это guid новой сущности (уникален в пределах таблицы TypedEntity);
2. TypeId – guid типа сущности (id из таблицы EntitySchema).

Дополнительно можно заполнить поля:

1. MetaInformation – это информация (типа JObject), связанная с сущностью на протяжении всей её жизни. Схема MetaInformation должна соответствовать схеме типа, указанного в поле TypeId;
2. AdditionalInformation – это информация, связанная с сущностью и дополняющая её. Является коллекцией IInformationHolder. Каждый элемент содержит TypeId (id схемы информации) и JObject (соответствующий этой схеме).

**Поля, заполняемые на стороне сервиса:**

1. LastModifiedDate;
2. CreatedDate;
3. Deleted;
4. Mandate/AccessGroups в соответствии со стратегией доступа и мандатом/группами пользователя.

**Проверки на стороне сервиса:**

1. Уникальность ключа Id;
2. Существование типа с ключом TypeId;
3. Соответствие MetaInformation типу с ключом TypeId;
4. Соответствие каждого блока IInformationHolder типу с ключом, указанным в IInformationHolder.

**Возвращаемый результат:**

ISuccessData содержит поле Succeeded типа bool, указывающее на успешность операции. True возвращается, только если все проверки пройдены успешно и сущность сохранена в БД.

**Ошибки:**

ServiceException. ErrorCode = 400.

* Если TypeId не существует;
* Если AdditionalInformation содержит null;
* Если один из блоков AdditionalInformation содержит дубликат передаваемой сущности;
* Если MetaInformation содержит невалидный json.

|  |
| --- |
| **После сохранения сущности генерируется CRUD-событие, на которое может реагировать, например, автопроцессинг.** |

### Обновление одной сущности

**Метод:**

Task<ISuccessData> UpdateEntity(ITypedEntityPatch patch, string sessionIdentifier);

**Параметры:**

* patch – набор изменений для сущности, в частности, изменений прав доступа и добавление/обновление/удаление метаданных;
* sessionIdentifier – идентификатор сессии клиента.

Списки метаданных для изменений не должны пересекаться между собой.

**Поведение:**

* Метод применяет к сущности изменения, указанные в patch;
* Каждое из изменений независимо от остальных, но частичное применение патча не допускается;
* Изменения метаинформации соответствуют по логике аналогичным из отдельных методов добавления/обновления/удаления метаданных;
* CRUD-события посылаются в шину соответственно произведённым изменениям (для каждого типа изменений отдельно).

**Возвращаемый результат:**

ISuccessData содержащий:

* Succeed  = true в случае успешного выполнения;
* Succeed  = false, ErrorData с информацией об ошибке.

**Ошибки:**

* 400 в случае некорректного запроса, например, при попытке обновления и удаления одной и той же метаинформации;
* 403 в случае отсутствия прав доступа к сущности;
* 404 в случае отсутствия сущности с указанным идентификатором;
* 408 в случае невозможности блокировки сущности для внесения изменений;
* 500 в иных случаях.
  1. Работа с метаинформацией в сущности

### Добавление метаинформации в сущность

Для добавления метаинформации в сущность порядок действий следующий:

1. Вызов метода AddInformationToEntityAsynс;
2. На стороне сервиса будет проведена проверка соответствия jsonInformation  схеме (infoTypeId- id схемы);
3. Будет проведена проверка уникальности id объекта jsonInformation в рамках AdditiotalInfromation и добавлено, если все ок, в AdditiotalInfromation, если infoTypeId не соответствует TypeId для сущности с идентификатором entityId. Если infoTypeId соответствует TypeId, то будет произведена проверка схемы и отсутствие metaInfromation в сущности entityId;
4. Если все ок, то будет произведено сохранение сущности и вызов CRUD-события в шине.

### Изменение метаинформации в сущности

Для изменения метаинформации в сущности порядок действий следующий:

1. Вызов метода UpdateEntityInformationAsync;
2. На стороне сервиса будет проведена проверка соответствия jsonInformation схеме (infoTypeId- id схемы);
3. Будет проведена проверка наличия объекта jsonInformation в AdditiotalInfromation и добавлено, если все ок, в AdditiotalInfromation, если infoTypeId не соответствует TypeId для сущности с идентификатором entityId. Если infoTypeId соответствует, TypeId то будет произведена проверка схемы и наличие metaInfromation в сущности entityId;
4. Если все ок, то будет произведено сохранение сущности и вызов CRUD события в шине.
   1. Операции с метаинформацией в нескольких сущностях

#### **Группа методов:**

Task<IResultWithErrorData<IReadOnlyCollection<IResultWithErrorData<IMetaData>>>> UpdateEntitiesInformationsAsync(IReadOnlyCollection<IMetaData> updates, string sessionIdentifier);

Task<IResultWithErrorData<IReadOnlyCollection<IResultWithErrorData<IMetaData>>>> AddEntitiesInformationsAsync(IReadOnlyCollection<IMetaData> updates, string sessionIdentifier);

Task<IResultWithErrorData<IReadOnlyCollection<IResultWithErrorData<IMetaDataType>>>> RemoveEntitiesMetaByTypeAsync(IReadOnlyCollection<IMetaDataType> updates, string sessionIdentifier);

Task<IResultWithErrorData<IReadOnlyCollection<IResultWithErrorData<IMetaDataIds>>>> RemoveEntitiesMetaByIdsAsync(IReadOnlyCollection<IMetaDataIds> updates, string sessionIdentifier);

**Параметры:**

* updates (список изменений в зависимости от метода);
* IMetaData – содержит тип метаданных, сами метаданные и идентификатор сущности, для которой нужно выполнить добавление/обновление указанных мета-данных;
* IMetaDataType – содержит тип метаданных и идентификатор сущности, из которой нужно выполнить удаление метаданных указанного типа;
* IMetaDataIds – содержит тип метаданных, список идентификаторов метаданных и идентификатор сущности, из которой нужно выполнить удаление указанных метаданных;
* sessionIdentifier – идентификатор сессии клиента.

**Поведение:**

Каждый из методов выполняет указанные операции. Логика выполнения аналогична логике операций над одной сущностью, включая рассылку CRUD-событий.

Списки могут содержать обновления как для одной сущности, так и для разных. Обновления выполняются посущностно, с предварительной группировкой из общего списка.

Методы допускают частичное выполнение обновлений.

**Возвращаемый результат:**

IResultWithErrorData<IReadOnlyCollection<IResultWithErrorData<IMetaDataType>>>

Результат содержит ошибки в случае их наличия.

В случае неуспешного обновления информация об ошибке содержится в поле ErrorData верхнего контейнера IResultWithErrorData. Это означает, что ни одно из изменений не выполнено.

В случае частичного применения обновлений список ошибок содержится в поле Result верхнего контейнера IResultWithErrorData. В списке могут содержаться как ошибки, относящиеся к одной сущности (соответственно ко всем обновлениям для этой сущности), так и ошибки, относящиеся к конкретному обновлению. В случае последнего в нижнем контейнере IResultWithErrorData содержится не только информация об ошибке, но и копия обновления, к которому она относится.

В случае успешного применения всех обновлений информация об ошибках отсутствует.

**Общие ошибки:**

* 408 в случае невозможности блокировки сущности для внесения изменений;
* 500 в иных случаях.

**Частные ошибки:**

* 400 в случае некорректного обновления (например, несоответствие типа или содержимого мета-информации схеме);
* 403 в случае отсутствия прав доступа к сущности;
* 404 в случае отсутствия сущности с указанным идентификатором;
  1. ****Другие методы****

### Проверка использования группы доступа в сущностях

#### **Метод:**

Task<ISuccessData> AccessGroupIsFree(Guid groupId, string sessionIdentifier);

#### **Параметры:**

* groupId - это идентификатор группы доступа;
* sessionIdentifier - идентификатор сессии клиента.

#### **Возвращаемый результат:**

ISuccessData, содержащий:

* Succeed = true - Если группа НЕ используется ни в одной сущности;
* Succeed = false - Если группа используется хотя бы в одной сущности;
* Succeed = false + ErrorData  - если произошла ошибка (код 500).

1. Processing
   1. IAeDetectorService

**(детектирование акустических событий)**

**Способ подключения:**

* Nuget-пакеты

|  |
| --- |
| **Внимание!**  Пользоваться нугет-пакетами нельзя до особого разрешения. Нужно цеплять прямую ссылку. |

* Platform-T.Processing.AeDetector.Service.Facade- реализация и интерфейс фасада;
* Platform-T.Processing.AeDetector.Service.Dto- дтошки;
* Platform-T.Processing.AeDetector.Service.Facade.Interfaces - интерфейс фасада.

**Конфигурация:**

"PT\_RABBIT\_MQ\_CONNECTION\_STRING": "postgres:wombat@<IP>:5672",

**Методы:**

Send()

Отправка запроса в TService

**Send:**

ISuccessData Send(IAeDetectorArguments aeDetectorArgs, string sessionId)

**Параметры:**

IAeDetectorArguments

public interface IAeDetectorArguments

{

Guid EntityId { get; }

string StorageKey { get; }

long FileLength { get; }

bool Forced { get; }

}

* EntityId ­­­−­­­­ Идентификатор Entity в Aggregation;
* StorageKey − ключ для StorageService;
* FileLength − длина файла в байтах;
* Forced − факт принудительной отправки запроса.

**Возвращаемое значение:**

ISuccessData

public interface DevTools.Errors.Core.Result.CommonResults.ISuccessData : IResultWithErrorData

{

bool Succeeded { get; }

}

**Принцип работы:**

Запрос отправляется по принципу fire-and-forget; на транспортном уровне используется IClientTransportManager.PostRequest.

То есть, ISuccessData содержит исключительно успешность отправки запроса в очередь.

Запрос обрабатывает processing-event-detector-dtt.

* 1. IFullTextIndexingService

**(инициализация результатов транскрибирования)**

Реализован Dtt-фасад к сервису предназначеного для инициализации результатов транскрибирования по entity.

* Интерфейс − IFullTextIndexingService.

**Способ подключения:**

Нужно цеплять прямую ссылку.

**Конфигурация:**

"PT\_RABBIT\_MQ\_CONNECTION\_STRING": "postgres:wombat@<IP>:5672",

**Методы:**

Send()

* Отправка запроса в TService.

**Send:**

ISuccessData Send(IFullTextIndexingArguments fullTextIndexingArgs, string sessionId)

**Параметры:**

IFullTextIndexingArguments

public interface IAeDetectorArguments

{

Guid EntityId { get; }

string StorageKey { get; }

long FileLength { get; }

string FileType { get; }

}

* EntityId − Идентификатор Entity в Aggregation;
* StorageKey − ключ для StorageService;
* FileLength − длина файла в байтах;
* FileType − тип файла.

**Возвращаемое значение:**

ISuccessData

public interface DevTools.Errors.Core.Result.CommonResults.ISuccessData : IResultWithErrorData

{

bool Succeeded { get; }

}

**Как работает:**

Запрос отправляется по принципу fire-and-forget, на транспортном уровне используется IClientTransportManager.PostRequest.

То есть, ISuccessData содержит исключительно успешность отправки запроса в очередь.

Запрос обрабатывает processing-full-text-indexing-dtt.

* 1. IIdentService

**(идентификация диктора по голосу)**

Реализован Dtt-фасад к сервису для идентификации диктора по голосу; позволяет строить голосовые модели дикторов и идентифицировать диктора в фонограмме.

Интерфейс − IIdentService.

**Способ подключения:**

* Nuget-пакеты:
* Platform-T.Processing.Ident.Service.Facade- реализация и интерфейс фасада;
* Platform-T.Processing.Ident.Requests.Dto- дтошки;
* Platform-T.Processing.Ident.Service.Facade.Interface - интерфейс фасада.

**Конфигурация:**

"PT\_RABBIT\_MQ\_CONNECTION\_STRING": "postgres:wombat@<IP>:5672",

**Методы:**

Send()

* Отправка запроса в TService.

**Send:**

ISuccessData Send(IIdentArguments identArgs, string sessionId)

**Параметры:**

IIdentArguments

public interface IIdentArguments

{

Guid EntityId { get; }

string StorageKey { get; }

long FileLength { get; }

Guid ChannelId { get; }

int JobPriority { get; }

bool Forced { get; }

}

* EntityId − Идентификатор Entity в Aggregation;
* StorageKey − ключ для StorageService;
* FileLength − длина файла в байтах;
* ChannelId − идентификатор для ChannelManager;
* JobPriority − приоритет задачи для TService;
* Forced − факт принудительной отправки запроса.

**Возвращаемое значение:**

ISuccessData

public interface DevTools.Errors.Core.Result.CommonResults.ISuccessData : IResultWithErrorData

{

bool Succeeded { get; }

}

**Принцип работы:**

Запрос отправляется по принципу fire-and-forget; на транспортном уровне используется IClientTransportManager.PostRequest.

Таким образом, ISuccessData содержит исключительно успешность отправки запроса в очередь. Запрос обрабатывает processing-ident-dtt.

* 1. ITranscriberService

**(распознавание речи)**

Реализован Dtt-фасад к сервису для распознования речи и звуков с переводом их в текст для физ. лиц. Интерфейс − ITranscriberService.

**Способ подключения:**

* Nuget-пакеты:
* Platform-T.Processing.Transcriber.Service.Facade- реализация и интерфейс фасада;
* Platform-T.Processing.Transcriber.Requests.Dto- дтошки;
* Platform-T.Processing.Transcriber.Service.Facade.Interfaces - интерфейс фасада.

|  |
| --- |
| **Внимание!**  Пользоваться нугет-пакетами нельзя до особого разрешения. Нужно цеплять прямую ссылку. |

**Конфигурация:**

"PT\_RABBIT\_MQ\_CONNECTION\_STRING": "postgres:wombat@<IP>:5672",

**в**

**Методы:**

Send()

* Отправка запроса в TService.

**Send:**

ISuccessData Send(ITranscriberArguments transcriberArgs, string sessionId)

**Параметры:**

ITranscriberArguments

public interface ITranscriberArguments

{

Guid EntityId { get; }

string StorageKey { get; }

Guid ChannelId { get; }

int JobPriority { get; }

bool Forced { get; }

Language Language { get; }

}

* EntityId − идентификатор Entity в Aggregation;
* StorageKey − ключ для StorageService;
* ChannelId − идентификатор для ChannelManager;
* JobPriority − приоритет задачи для TService;
* Forced − факт принудительной отправки запроса;
* Language − выбранный язык; если он не указан, применяется мультиязычное или обычное транскрибирование в зависимости от настроек сервиса.

**Возвращаемое значение:**

ISuccessData

public interface DevTools.Errors.Core.Result.CommonResults.ISuccessData : IResultWithErrorData

{

bool Succeeded { get; }

}

**Принцип работы:**

Запрос отправляется по принципу fire-and-forget; на транспортном уровне используется IClientTransportManager.PostRequest.

Таким образом, ISuccessData содержит исключительно успешность отправки запроса в очередь. Запрос обрабатывает processing-transcriber-dtt.

1. ArchiveSearchQueriesEngineModule

**(модуль движка АПЗ)**

* 1. IASQRegistry API

### RegisterSearchQueryAsync

**(регистрация АПЗ на выполнение)**

**IN**

| **Аргумент** | **Тип** | **Комментарий** |
| --- | --- | --- |
| searchQueryId | Guid | Идентификатор поискового запроса по архиву (ASQ | АПЗ) |
| sessionIdentifier | String | Идентификатор сессии. |

**OUT**

| **Аргумент** | **Тип** | **Комментарий** |
| --- | --- | --- |
| Result | ISuccessData | Операция считается проведенной **успешно**, если в экземпляре ISuccessData, который вернул метод:   * В поле **ErrorData**содержится **null**. В противном случае оно будет содержать код и описание ошибки; * В поле **Succeeded** значении **true.**   Операция считается проведенной **провально**, если в экземпляре IResultWithErrorData, который вернул метод:   * В поле **ErrorData**содержится **код**и **описание ошибки**. (в поле **Succeeded**значение **false).** |

### CancelSearchQueryAsync

**(отмена выполнения АПЗ)**

**IN**

| **Аргумент** | **Тип** | **Комментарий** |
| --- | --- | --- |
| searchQueryId | Guid | Идентификатор поискового запроса по архиву (ASQ | АПЗ) |
| sessionIdentifier | String | Идентификатор сессии. |

**OUT**

| **Аргумент** | **Тип** | **Комментарий** |
| --- | --- | --- |
| Result | ISuccessData | Операция считается проведенной **успешно**, если в экземпляре ISuccessData, который вернул метод:   * В **ErrorData**содержится **null**. В противном случае оно будет содержать код и описание ошибки; * в поле **Succeeded** значении **true.**   Операция считается проведенной **провально**, если в экземпляре IResultWithErrorData, который вернул метод:   * В поле **ErrorData**содержится **код**и **описание ошибки**. (в поле **Succeeded**значение **false).** |

* 1. REST API

### Тип 1

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400/api/v1/asqe-engine/{id1}/register-sq |
| **Описание:** | Регистрация поискового запроса в БД сервиса поисковых запросов по архиву |
| **Пример ответа:** | Стандартный 200 или ошибка |
| **Необходимые права (Permissions):** | ArchiveSearchQueryEnginePermissions.RegisterSearchQueryInEngine |
| **Примечание:** | * Поисковый запрос с переданным id уже зарегистрирован − код ошибки 400; * Поисковый запрос с переданным id не зарегистрирован в модуле поисковых запросов − код ошибки 400; * Поисковый запрос с переданным id зарегистрирован в модуле поисковых запросов не как 'Archive' − код ошибки 400. |

### ****Тип 2****

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400/api/v1/asqe-engine/{id1}/cancel-sq |
| **Описание:** | Отмена зарегистрированного поискового запроса |
| **Пример ответа:** | Cтандартный 200 или ошибка |
| **Необходимые права (Permissions):** | ArchiveSearchQueryEnginePermissions.RegisterSearchQueryInEngine |
| **Примечание:** | Поисковый запрос с переданным id еще не был зарегистрирован – код ошибки 400. |

### Тип 3

|  |  |
| --- | --- |
| **Название и тип:** | GET |
| **Путь:** | {host}:5400/api/v1/asq-engine/statistics/get-asq-statistics |
| **Описание:** | Получение статистики по АПЗ |
| **Пример ответа:** | Cтандартный 200 или ошибка |
| **Необходимые права (Permissions):** | ArchiveSearchQueryEnginePermissions.RegisterSearchQueryInEngine |
| **Примечание:** | id – идентификатор поискового запроса. |

В случае успешного выполнения запроса будет получен ответ, содержащий следующую информацию:

{  
    "totalCountAsqs": 25, // общее число ПЗ в движке АПЗ  
    "activeAsqs": 5, // число ПЗ в состоянии Active  
    "postprocessingAsqs": 5, // число ПЗ в состоянии Postprocessing  
    "completedAsqs": 15, // число ПЗ в состоянии Completed  
    "totalCountResults": 38, // число результатов ПЗ в движке АПЗ  
    "totalCountVoiceIdentResults": 2 // число результатов поиска по голосу в движке АПЗ  
}

1. SearchQueriesModule

**(модуль движка ППЗ)**

* 1. [API SearchQueriesController.ChangeNameOfSearchQueryAsync](https://confluence.speechpro.com/display/NEZABUDKASORM/API+SearchQueriesController.ChangeNameOfSearchQueryAsync)

### Общая информация

Метод SearchQueriesController.ChangeNameOfSearchQueryAsync позволяет изменить наименование поискового запроса с указанным идентификатором.

При изменении наименования ПЗ производится отправка CRUD-сообщения, содержащего обновленную модель или IArchiveSearchQuery (для АПЗ), либо IProcessingSearchQuery (для ППЗ).

В случае изменения наименования ПЗ на аналогичное текущему, CRUD-сообщение не отправляется.

| **Тип** | **Путь** | **Необходимые Permissions** |
| --- | --- | --- |
| POST | {host}:5400/api/v1/search-queries/change-sq-name | * SearchQueryPermissions.**SqChangeNameOfSearchQuery** |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Также ограничения налагаются модулем аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | SearchQueryFilterDto filterParams |
| **Описание:** | Параметры фильтра для получения поисковых запросов |
| **Ограничения, накладываемые контроллером:** | Ограничений не налагается |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<Guid>* Ids |
| **Описание:** | Коллекция идентификаторов фильтров, над которыми выполняется последующая фильтрация |
| **Ограничения, накладываемые контроллером:** | Ограничений не налагается |
| **Примечание:** | Если список пустой, или null, поиск осуществляется по всем поисковым запросам |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *StringFilterDto* NameFilter |
| **Описание:** | Шаблон наименования. |
| **Ограничения, накладываемые контроллером:** | Ограничений не налагается |
| **Примечание:** | Если null, поиск осуществляется по всем поисковым запросам |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *string* Value |
| **Описание:** | Значение, по которому осуществляется поиск |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *StringPredicate* Predicate |
| **Описание:** | Предикат. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Возможные значения StringPredicate:   * Equal; * Contains; * StartWith. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<Guid>* FilterTemplateIds |
| **Описание:** | Идентификаторы шаблонов-фильтров |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой, или null, поиск осуществляется без учета наличия/отсутствия ссылки на шаблон-фильтр. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *bool?* ContainsFilterTemplateReference |
| **Описание:** | Признак, что ищутся запросы, содержащие ссылку на шаблон-фильтр, либо не содержащие таковой. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если null – наличие или отсутствия ссылки не учитывается. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *QueryType?* QueryType |
| **Описание:** | Тип запроса. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если null, то не учитывается при фильтрации.  Возможные значения QueryType:   * Archive; * Processing. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *QueryType?* QueryType |
| **Описание:** | Тип запроса |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если null, то не учитывается при фильтрации.  Возможные значения QueryType:   * Archive; * Processing. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<SearchQueryStatus>* Statuses |
| **Описание:** | Состояния запросов |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или null, то не учитывается в фильтрации.  Возможные значения SearchQueryStatus:   * Active; * NotActive; * Completed; * CompletedWithError; * Deleted. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<Guid>* CreatedBy |
| **Описание:** | Список идентификаторов пользователей, которые создали запрос. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или null, то не учитывается в фильтрации. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<Guid>* UpdatedBy |
| **Описание:** | Список идентификаторов пользователей, которые изменили запрос |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или null, то не учитывается в фильтрации. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *DateTimeOffsetRangeDto* CreatedAtRange |
| **Описание:** | Временной промежуток, когда был создан запрос (From..To). |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Интервал может быть открытым, например **From** – задан, а **To** – null, и наоборот. Не учитывается, если параметр равен **null** или если **From** и **To** равны **null**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *DateTimeOffsetRangeDto* UpdatedAtRange |
| **Описание:** | Временной промежуток, когда был изменен запрос (From..To). |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Интервал может быть открытым, например **From** – задан, а **To** – null и наоборот. Не учитывается, если параметр равен **null** или если **From** и **To** равны **null**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *bool? SaveResults* |
| **Описание:** | Признак, надо ли сохранять результаты; имеет значение только в рамках запроса для архивного поиска. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Интервал может быть открытым, например **From** – задан, а **To** – null и наоборот. Не учитывается, если параметр равен **null** или если **From** и **To** равны **null**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *int?* Mandate |
| **Описание:** | Мандат – как и у ОУ, на данный момент это просто информационное поле. |
| **Ограничения, накладываемые контроллером:** | Ограничений не налагается |
| **Примечание:** | Если равен null, то не учитывается. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *int* Skip |
| **Описание:** | Количество пропускаемых элементов |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Не учитывается при подсчете ПЗ, соответствующих фильтру. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *int?* Limit |
| **Описание:** | Наибольшее количество элементов, содержащихся в результирующей коллекции. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Не учитывается при подсчете ПЗ, соответствующих фильтру. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *bool* SearchAmongDeleted |
| **Описание:** | Признак необходимости поиска в числе удаленных поисковых запросов; имеет значение только в рамках запроса для архивного поиска. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *ExpiryPeriodRangeDto* ExpiryPeriodRange |
| **Описание:** | Интервал фильтрации по периоду хранения (From..To), в днях |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *DateTimeOffsetRangeDto* CompletionDateRange |
| **Описание:** | Временной промежуток, когда был завершен запрос (From..To). |
| **Ограничения, накладываемые контроллером:** | From <= To |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *ActionFilterDto* ActionFilter |
| **Описание:** | Фильтр по экшену на результатах запроса (наличие или отсутствие определенного типа) |
| **Ограничения, накладываемые контроллером:** | Тип экшена должен входить в перечень допустимых |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<long>* RecipientsIds |
| **Описание:** | Фильтр по пользователям, которых будут уведомлять в экшне нотификации |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *bool* SortAsc |
| **Описание:** | Признак того, что сортировка должна производиться по возрастанию |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Значение по умолчанию: **false** |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *SortableFieldName* SortByFieldName |
| **Описание:** | Указатель на наименование свойства ПЗ, по которому будет производиться сортировка |
| **Ограничения, накладываемые контроллером:** | Значение должно существовать в перечислимом типе |
| **Примечание:** | Значение по умолчанию: **ByCreationDate**.Возможные значения **SortableFieldName**:   * ByCreationDate; * ByName; * ByStatus; * ByUpdateDate; * ByCompleteDate. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *SortableFieldName* SortByFieldName |
| **Описание:** | Указатель на наименование свойства ПЗ, по которому будет производиться сортировка. |
| **Ограничения, накладываемые контроллером:** | Значение должно существовать в перечислимом типе. |
| **Примечание:** | Значение по умолчанию: **ByCreationDate**.Возможные значения **SortableFieldName**:   * ByCreationDate; * ByName; * ByStatus; * ByUpdateDate; * ByCompleteDate. |

|  |  |
| --- | --- |
| **Название и тип:** | CancellationToken token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Операция считается проведенной успешно, если в результате, который вернул метод:

* Код ответа: **200**;
* Находится количество поисковых запросов (long), попадающих под условия фильтра.

Операция считается проведенной провально, если в результате, который вернул метод:

* Код ответа: код ошибки;
* Находится экземпляр ErrorData, содержащий код и описание ошибки.

Варианты содержания результата, который возвращает метод, взаимоисключающие, т. е. всегда возвращается один из них: либо количество ПЗ, либо информация об ошибке.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации. |

* 1. API SearchQueriesController.DeleteBatchSearchQueryAsync

### Общая информация

Метод SearchQueriesController.DeleteBatchSearchQueryAsync позволяет удалить поисковые запросы с указанными идентификаторами.

Если во время удаления одного из поисковых запросов произошла ошибка или запрос невозможно удалить, то происходит логирование с уровнем Warn. Метод в этом случае ошибку не возвращает, продолжает работу по удалению остальных запросов и результат выполнения будет считаться успешным.

**Для поисковых запросов по архиву:**

При удалении АПЗ также удаляются относящиеся к нему Action'ы, но CRUD-сообщение отправляется только об удалении АПЗ.

**Для поисковых запросов по потоку:**

При удалении ППЗ отправляется CRUD-сообщение, содержащие Id удаленного ППЗ.

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400//api/v1/search-queries/delete-batch-sq |
| Необходимые permissions: | SearchQueryPermissions.**SqDeleteSearchQuery** |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Ограничения также налагаются модулем Аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | *ByIdsParamsDto* paramsDto |
| **Описание:** | Параметры поискового запроса для удаления |
| **Ограничения, накладываемые контроллером:** | Не должен быть равен null |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  Guid Ids |
| **Описание:** | Идентификаторы поисковых запросов |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | *CancellationToken* token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Task<ISuccessData>

Операция считается успешной, если в экземпляре ISuccessData, который вернул метод:

* В поле **Succeeded** находится значение **true**;
* В поле **ErrorData** содержится **null**.

Операция считается неуспешной, если в экземпляре ISuccessData, который вернул метод:

* Поле Succeeded равно false;
* Поле ErrorData содержит код и описание ошибки.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ. |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса. |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации |

* 1. API SearchQueriesController.DeleteSearchQueryAsync

### Общая информация

Метод SearchQueriesController.DeleteSearchQueryAsync позволяет удалить поисковый запрос с указанным идентификатором.

В случае, если поисковый запрос ещё выполняется в момент запроса на удаление, выполнение поискового запроса отменяется и происходит его удаление.

**Для поисковых запросов по архиву:**

При удалении АПЗ также удаляются относящиеся к нему Action'ы, но CRUD-сообщение отправляется только об удалении АПЗ.

**Для поисковых запросов по потоку:**

При удалении ППЗ отправляется CRUD-сообщение, содержащее Id удаленного ППЗ.

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400//api/v1/search-queries/delete-sq |
| Необходимые permissions: | SearchQueryPermissions.**SqDeleteSearchQuery** |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Ограничения также налагаются модулем Аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | *ByIdParamsDto* paramsDto |
| **Описание:** | Параметры поискового запроса для удаления |
| **Ограничения, накладываемые контроллером:** | Не должен быть равен null |
| **Примечание:** |  |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  Guid Id |
| **Описание:** | Идентификатор поискового запроса |
| **Ограничения, накладываемые контроллером:** | Поисковый запрос с таким идентификатором должен существовать в системе или не иметь статуса Deleted (в случае ППЗ). |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | *CancellationToken* token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Task<ISuccessData>

Операция считается проведенной **успешно**, если в экземпляре ISuccessData, который вернул метод:

* В поле **Succeeded** находится значение **true**;
* В поле **ErrorData** содержится **null**.

Операция считается проведенной **провально**, если в экземпляре ISuccessData, который вернул метод:

* Поле Succeeded равно false;
* Поле ErrorData содержит код и описание ошибки.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса. |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации. |

* 1. API SearchQueriesController.GetMaxCountBySqTypeAsync

### Общая информация

Метод SearchQueriesController.GetMaxCountBySqTypeAsync – максимальное количество разрешенных поисковых запросов в системе для каждого типа запросов (АПЗ или ППЗ).

|  |  |
| --- | --- |
| **Название и тип:** | GET |
| **Путь:** | {host}:5400/api/v1/search-queries/get-max-search-queries-count/{queryType} |
| Необходимые permissions: | SearchQueryPermissions.SqGetCount |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Ограничения также налагаются модулем Аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | *string* QueryType |
| **Описание:** | Тип поискового запроса |
| **Ограничения, накладываемые контроллером:** | Допустимые значения:   * Archive; * Processing. |
| **Примечание:** |  |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  Guid Id |
| **Описание:** | Идентификатор поискового запроса |
| **Ограничения, накладываемые контроллером:** | Поисковый запрос с таким идентификатором должен существовать в системе или не иметь статуса Deleted (в случае ППЗ). |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | *CancellationToken* token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Task<IResultWithErrorData<int>

Операция считается проведенной **успешно**, если в экземпляре IResultWithErrorData, который вернул метод:

* В поле **Result** находится максимальное количество запросов для заданного типа;
* В поле **ErrorData** содержится **null**.

Операция считается проведенной провально, если в экземпляре IResultWithErrorData, который вернул метод:

* Поле **Result** равно **null**;
* Поле ErrorData содержит **код и описание ошибки**.

Поля **Result** и **ErrorData** экземпляра IResultWithErrorData, который возвращает метод, взаимоисключающие, т.е. всегда заполнено из них только одно, второе при этом имеет значение **null**. Одновременно равняться **null** оба поля не могут.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ. |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса. |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации. |

* 1. API SearchQueriesController.GetSearchQueriesByFilterAsync

### Общая информация

Метод SearchQueriesController.GetSearchQueriesByFilterAsync позволяет получить коллекцию поисковых запросов, удовлетворяющих переданным параметрам фильтра с возможностью пропустить и взять нужное их количество.

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400/api/v1/search-queries/get-sq-by-filter |
| Необходимые permissions: | SearchQueryPermissions.**SqGetSearchQueriesByFilter** |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Также ограничения налагаются модулем Аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | *SearchQueryFilterDto* filterParams |
| **Описание:** | Параметры фильтра для получения поисковых запросов |
| **Ограничения, накладываемые контроллером:** | Не может быть null. |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection*<*Guid*> Ids |
| **Описание:** | Коллекция идентификаторов фильтров, над которыми выполняется последующая фильтрация |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или **null**, поиск осуществляется по всем поисковым запросам. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *StringFilterDto* NameFilter |
| **Описание:** | Шаблон наименования |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если null, поиск осуществляется по всем поисковым запросам. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *string* Value |
| **Описание:** | Значение, по которому осуществляется поиск |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *StringPredicate* Predicate |
| **Описание:** | Предикат |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Возможные значения StringPredicate:   * Equal; * Contains; * StartWith. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection*<*Guid*> FilterTemplateIds |
| **Описание:** | Идентификаторы шаблонов-фильтров |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или **null**, поиск осуществляется без учета наличия/отсутствия ссылки на шаблон-фильтр. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *bool*? ContainsFilterTemplateReference |
| **Описание:** | Признак, что ищутся запросы, содержащие ссылку на шаблон-фильтр, либо не содержащие таковой. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если **null** – наличие или отсутствия ссылки не учитывается. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *QueryType?* QueryType |
| **Описание:** | Тип запроса |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если null, то не учитывается при фильтрации. Возможные значения QueryType:   * Archive; * Processing. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<SearchQueryStatus>* Statuses |
| **Описание:** | Состояния запросов. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или null, то не учитывается в фильтрации. Возможные значения SearchQueryStatus:   * Active: * NotActive; * Completed; * CompletedWithError; * Deleted. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *TargetSystem?* TargetSystemFilter |
| **Описание:** | Целевая система поиска |
| **Ограничения, накладываемые контроллером:** | Значение должно существовать в перечислимом типе |
| **Примечание:** | Если список пустой или **null**, то не учитывается в фильтрации. Возможные значения TargetSystemFilter:   * Aggregation; * Analysis. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<Guid>* CreatedBy |
| **Описание:** | Список идентификаторов пользователей, которые создали запрос |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или null, то не учитывается в фильтрации. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<Guid>* UpdatedBy |
| **Описание:** | Список идентификаторов пользователей, которые изменили запрос |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если список пустой или null, то не учитывается в фильтрации. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *DateTimeOffsetRangeDto* CreatedAtRange |
| **Описание:** | Временной промежуток когда был создан запрос (From..To) |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Интервал может быть открытым, например From − задан, а To − null и наоборот. Не учитывается, если параметр равен **null** или если From и To равны **null**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *DateTimeOffsetRangeDto* UpdatedAtRange |
| **Описание:** | Временной промежуток когда был изменен запрос (From..To) |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Интервал может быть открытым, например From − задан, а To − **null** и наоборот. Не учитывается, если параметр равен **null** или если From и To равны **null**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *bool?* SaveResults |
| **Описание:** | Признак, надо ли сохранять результаты, имеет значение только в рамках запроса для архивного поиска. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если равен **null**, не учитывается в фильтрации. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *int?* Mandate |
| **Описание:** | Мандат – как и у ОУ, на данный момент это просто информационное поле. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Если равен **null**, то не учитывается. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  intSkip |
| **Описание:** | Количество пропускаемых элементов |
| **Ограничения, накладываемые контроллером:** | Должно быть ≥ 0. |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  int? Limit |
| **Описание:** | Наибольшее количество элементов, содержащихся в результирующей коллекции |
| **Ограничения, накладываемые контроллером:** | Должно быть ≥ 1. |
| **Примечание:** | Если null, используется значение по умолчанию (100). |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  bool SearchAmongDeleted |
| **Описание:** | Признак необходимости поиска в числе удаленных поисковых запросов; имеет значение только в рамках запроса для архивного поиска. |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  bool SortAsc |
| **Описание:** | Признак того, что сортировка должна производиться по возрастанию |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Значение по умолчанию: **false**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *SortableFieldName* SortByFieldName |
| **Описание:** | Указатель на наименование свойства ПЗ, по которому будет производиться сортировка |
| **Ограничения, накладываемые контроллером:** | Значение должно существовать в перечислимом типе |
| **Примечание:** | Значение по умолчанию: ByCreationDate. Возможные значения SortableFieldName:   * ByCreationDate; * ByName; * ByStatus; * ByUpdateDate; * ByCompleteDate. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *ExpiryPeriodRangeDto* ExpiryPeriodRange |
| **Описание:** | Интервал фильтрации по периоду хранения (From..To) |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | Интервал может быть открытым, например From – задан, а To – null и наоборот. Не учитывается, если параметр равен **null** или если From и To равны **null**. |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *DateTimeOffsetRangeDto* CompletionDateRange |
| **Описание:** | Временной промежуток, когда был завершен запрос (From..To) |
| **Ограничения, накладываемые контроллером:** | From ≤ To. |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *ActionFilterDto* ActionFilter |
| **Описание:** | Фильтр по экшену на результатах запроса (наличие или отсутствие определенного типа) |
| **Ограничения, накладываемые контроллером:** | Тип экшена должен входить в перечень допустимых |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *IReadOnlyCollection<long>* RecipientsIds |
| **Описание:** | Фильтр по пользователям, которых будут уведомлять в экшне нотификации |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  *CancellationToken* token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Task<IResultWithErrorData<Array<SearchQueryDto>>>

Операция считается проведенной **успешно**, если в экземпляре IResultWithErrorData, который вернул метод:

* В поле **Result** находится коллекция, попадающих под условия фильтра, поисковых запросов **Array**<**SearchQueryDto**>;
* В поле **ErrorData** содержится **null**.

Операция считается проведенной **провально**, если в экземпляре IResultWithErrorData, который вернул метод:

* Поле **Result** равно **null**;
* Поле ErrorData содержит код и описание ошибки.

Поля **Result** и **ErrorData**экземпляра IResultWithErrorData, который возвращает метод, взаимоисключающие, т. е. всегда заполнено из них только одно, второе при этом имеет значение **null**. Одновременно равняться **null** оба поля не могут.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса. |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации. |

* 1. API SearchQueriesController.GetSearchQueryByIdAsync

### Общая информация

Метод SearchQueriesController.GetSearchQueryByIdAsync позволяет получить поисковый запрос с указанным идентификатором.

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400/api/v1/search-queries/get-sq-by-id |
| Необходимые permissions: | SearchQueryPermissions.**SqGetSearchQueryById** |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Также, ограничения налагаются модулем Аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | *ByIdParamsDto* paramsDto |
| **Описание:** | Параметры поискового запроса для удаления |
| **Ограничения, накладываемые контроллером:** | Ограничений не налагается |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | ↳  Guid Id |
| **Описание:** | Идентификатор поискового запроса |
| **Ограничения, накладываемые контроллером:** | Ограничений не налагается |
| **Примечание:** | - |

|  |  |
| --- | --- |
| **Название и тип:** | *CancellationToken* token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Task<IResultWithErrorData<SearchQueryDto>>

Операция считается проведенной **успешно**, если в экземпляре IResultWithErrorData, который вернул метод:

* В поле Result находится поисковый запрос SearchQueryDto;
* В поле **ErrorData** содержится **null**.

Операция считается проведенной **провально**, если в экземпляре IResultWithErrorData, который вернул метод:

* Поле **Result** равно **null**;
* Поле ErrorData содержит код и описание ошибки.

Поля **Result** и **ErrorData**экземпляра IResultWithErrorData, который возвращает метод, взаимоисключающие, т. е. всегда заполнено из них только одно, второе при этом имеет значение **null**. Одновременно равняться **null** оба поля не могут.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ. |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса. |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации. |

* 1. API SearchQueriesController.TryMarkAsClearedAsync
  2. Общая информация

Метод SearchQueriesController.TryMarkAsClearedAsync предварительной маркировки АПЗ в МПЗ как «зачищенный». Возвращает успешно промаркированные идентификаторы АПЗ.

|  |  |
| --- | --- |
| **Название и тип:** | POST |
| **Путь:** | {host}:5400/api/v1/search-queries/mark-asq-as-cleared |
| Необходимые permissions: | SearchQueryPermissions.**SqChangeAsqExpirationDate** |

### Параметры

|  |  |
| --- | --- |
| **Название и тип:** | *string* authorizationString |
| **Описание:** | - |
| **Ограничения, накладываемые контроллером:** | * Схема авторизации: Bearer; * Также, ограничения налагаются модулем Аутентификации. |
| **Примечание:** | [FromHeader(Name = "Authorization")] |

|  |  |
| --- | --- |
| **Название и тип:** | *IReadOnlyCollection<Guid>* sqIds |
| **Описание:** | Набор идентификаторов АПЗ, подлежащие маркировке как «зачищенные» |
| **Ограничения, накладываемые контроллером:** | * Не может быть null; * Должен содержать элементы в коллекции; * Не должен содержать Guid.Empty. |
| **Примечание:** | Если передать Guid ранее промаркированного АПЗ, то он будет возвращен в результирующей коллекции. |

|  |  |
| --- | --- |
| **Название и тип:** | *CancellationToken* token |
| **Описание:** | Токен для отмены операции |
| **Ограничения, накладываемые контроллером:** | - |
| **Примечание:** | - |

### Возвращаемое значение

Task<IResultWithErrorData<IReadOnlyCollection<Guid>>>

Операция считается проведенной **успешно**, если в экземпляре IResultWithErrorData, который вернул метод:

* В поле **Result** находится коллекция попадающих под условия фильтра поисковых запросов **<IReadOnlyCollection<Guid>>**;
* В поле **ErrorData** содержится **null**.

Операция считается проведенной **провально**, если в экземпляре IResultWithErrorData, который вернул метод:

* Поле **Result** равно **null**;
* Поле ErrorData содержит код и описание ошибки.

Поля **Result** и **ErrorData**экземпляра IResultWithErrorData, который возвращает метод, взаимоисключающие, т. е. всегда заполнено из них только одно, второе при этом имеет значение **null**. Одновременно равняться **null** оба поля не могут.

### Возвращаемые ошибки

|  |  |  |
| --- | --- | --- |
| **Код** | **Краткое описание** | **Случай возникновения** |
| 400 | Ошибка валидации входных параметров метода, кроме sessionIdentifier | Возникает в случае, когда входные параметры не удовлетворяют описанным выше ограничениям на аргументы метода. Возвращается при возникновении первого не прошедшего ограничения. |
| 499 | Отмена запроса | Возникает при отмене запроса на получение истории ППЗ |
| 500 | Непредвиденная ошибка | Непредвиденная ошибка при обработке запроса на стороне клиента или сервиса. |
| Разные | Ошибка валидации параметра authorizationString | Определяются и описываются модулем Аутентификации. |

1. AccountingModule

**(модуль системы учета)**

* 1. Голосовая информация

Голосовая информация может содержать как просто аудиофайлы, так и голосовую модель, построенную на основе этих файлов.

### IVoiceInfoData

|  |  |
| --- | --- |
| **Поле:** | VoiceModelData |
| **Тип:** | IVoiceModelData |
| **Описание:** | Содержит информацию о голосовой модели |
| **Ограничения:** | Должна быть **null**, если нет хотя бы одного файла с сегментацией. |

|  |  |
| --- | --- |
| **Поле:** | AudioFileWithSegmentationDatas |
| **Тип:** | IReadOnlyCollection<IAudioFileWithSegmentationData> |
| **Описание:** | Содержит информацию об аудиофайлах |
| **Ограничения:** | * Не может содержать null в качестве элемента коллекции; * Не может равняться null, если IVoiceInfoData != null; * Не может быть пустой коллекцией. |

public interface IVoiceInfoData

{

    IVoiceModelData VoiceModelData { get; }

    IReadOnlyCollection<IAudioFileWithSegmentationData> AudioFileWithSegmentationDatas { get; }

}

### IVoiceModelData

|  |  |
| --- | --- |
| **Поле:** | VoiceModelKey |
| **Тип:** | string |
| **Описание:** | Ключ; на данный момент это путь хранения голосовой модели внутри TService |
| **Ограничения:** | 1. Не может принимать значение **null**, **empty** или состоять только из пробельных символов, если IVoiceModelData != null. |
| **Примечание:** | 1. Значение ключа должно заканчиваться на ".vm". В противном случае сервис **Ident вернет ошибку 500**. |

|  |  |
| --- | --- |
| **Поле:** | VoiceModelMetaData |
| **Тип:** | IVoiceModelMetaData |
| **Описание:** | Метаинформация по голосовой модели |
| **Ограничения:** | 1. Не может принимать значение **null**, если IVoiceModelData != null. |
| **Примечание:** | 1. Значение ключа должно заканчиваться на ".vm". В противном случае сервис **Ident вернет ошибку 500**. |

public interface IVoiceModelData

{

    string VoiceModelKey { get; }

    IVoiceModelMetaData VoiceModelMetaData { get; }

}

### IVoiceModelMetaData

|  |  |
| --- | --- |
| **Поле:** | State |
| **Тип:** | VoiceModelState |
| **Описание:** | Состояние голосовой модели |
| **Ограничения:** | Может принимать только значения, соответствующие элементам перечислимого типа:   * Normal; * Partial.   Остальные VoiceModelState являются состояниями ошибки. Модели с ними не могут сохраняться в Accounting модуля. Вернётся ошибка 400. |

|  |  |
| --- | --- |
| **Поле:** | BioSdkVersion |
| **Тип:** | string |
| **Описание:** | Версия Sdk |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов. |

public interface IVoiceModelMetaData

{

    VoiceModelState State { get; }

    string BioSdkVersion { get; }

}

### VoiceModelState

| **Значение** | **Описание** |
| --- | --- |
| Normal | Нормальная |
| NotEnoughData | Недостаточно чистой речи |
| Partial | Частичная модель - когда модель создается из очень длительного файла и создается из-за этого только по части файла |
| Error | Ошибка |

На данный момент NotEnoughData и Error являются состояниями ошибки и такие модели не должны сохранятся в Учетной системе.

public enum VoiceModelState

{

    Normal,

    NotEnoughData,

    Partial,

    Error

}

### IAudioFileWithSegmentationData

|  |  |
| --- | --- |
| **Поле:** | FileKey |
| **Тип:** | string |
| **Описание:** | Ключ файла; на данный момент это ключ хранения файла на Storage |
| **Ограничения:** | * Не может принимать значение **null**, **empty** или состоять только из пробельных символов; * Уникально внутри коллекции AudioFileWithSegmentationDatas. |

|  |  |
| --- | --- |
| **Поле:** | AudioMetaData |
| **Тип:** | IAudioMetaData |
| **Описание:** | Аудио метаинформация |
| **Ограничения:** | Не может принимать значение **null**. |

|  |  |
| --- | --- |
| **Поле:** | FileMetaData |
| **Тип:** | IFileMetaData |
| **Описание:** | Файловая метаинформация |
| **Ограничения:** | Не может принимать значение **null**. |

|  |  |
| --- | --- |
| **Поле:** | SegmentationData |
| **Тип:** | ISegmentationData |
| **Описание:** | Информация о сегментации |
| **Ограничения:** | – |

SegmentationData может отсутствовать – это будет означать, что этот файл не участвовал в построении голосовой модели.

public interface IAudioFileWithSegmentationData

{

    string FileKey { get; }

    IAudioMetaData AudioMetaData { get; }

    IFileMetaData FileMetaData { get; }

    ISegmentationData SegmentationData { get; }

}

### IAudioMetaData

|  |  |
| --- | --- |
| **Поле:** | Comment |
| **Тип:** | string |
| **Описание:** | Комментарий к аудио файлу |

|  |  |
| --- | --- |
| **Поле:** | Duration |
| **Тип:** | TimeSpan |
| **Описание:** | Длительность файла |

|  |  |
| --- | --- |
| **Поле:** | ChannelsCount |
| **Тип:** | int |
| **Описание:** | Количество аудиоканалов |

|  |  |
| --- | --- |
| **Поле:** | StreamNumber |
| **Тип:** | int |
| **Описание:** | Номер аудиострима |

|  |  |
| --- | --- |
| **Поле:** | SampleRate |
| **Тип:** | int? |
| **Описание:** | Частота дискретизации |

|  |  |
| --- | --- |
| **Поле:** | CodecName |
| **Тип:** | string |
| **Описание:** | Название кодека |

|  |  |
| --- | --- |
| **Поле:** | CodecBitRate |
| **Тип:** | int? |
| **Описание:** | Битрейт кодека |

public interface IAudioMetaData

{

    string Comment { get; }

    TimeSpan Duration { get; }

    int ChannelsCount { get; }

    int StreamNumber { get; }

    int? SampleRate { get; }

    string CodecName { get; }

    int? CodecBitRate { get; }

}

### IFileMetaData

|  |  |
| --- | --- |
| **Поле:** | AddedAt |
| **Тип:** | DateTimeOffset |
| **Описание:** | Время добавление файла в UTC |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | CreatedByUserIdentifier |
| **Тип:** | object |
| **Описание:** | Идентификатор пользователя, добавившего файл. На данный момент это Guid − учетный идентификатор пользователя. |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | FileName |
| **Тип:** | string |
| **Описание:** | - |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов. |

|  |  |
| --- | --- |
| **Поле:** | FileExtension |
| **Тип:** | string |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | BinHash |
| **Тип:** | string |
| **Описание:** | Бинарный хэш файла |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов. |

public interface IFileMetaData

{

    DateTimeOffset AddedAt { get; }

    object CreatedByUserIdentifier { get; }

    string FileName { get; }

    string FileExtension { get; }

    string BinHash { get; }

}

### ISegmentationData

|  |  |
| --- | --- |
| **Поле:** | SegmentationKey |
| **Тип:** | string |
| **Описание:** | Ключ; на данный момент это путь хранения голосовой сегментации по файлу внутри TService |
| **Ограничения:** | 1. Не может принимать значение **null**, **empty** или состоять только из пробельных символов. Уникально внутри коллекции AudioFileWithSegmentationDatas. |

|  |  |
| --- | --- |
| **Поле:** | SegmentationMetaData |
| **Тип:** | ISegmentationMetaData |
| **Описание:** | Метаинформация по сегментации по файлу |
| **Ограничения:** | Не может принимать значение **null**. |

public interface ISegmentationData

{

    string SegmentationKey { get; }

    ISegmentationMetaData SegmentationMetaData { get; }

}

### ISegmentationMetaData

|  |  |
| --- | --- |
| **Поле:** | ChannelNumber |
| **Тип:** | ChannelNumber |
| **Описание:** | Номер канала |
| **Ограничения:** | Может принимать только значения, соответствующие элементам перечислимого типа. |

|  |  |
| --- | --- |
| **Поле:** | ChannelType |
| **Тип:** | ChannelType |
| **Описание:** | Тип аудио канала |
| **Ограничения:** | Может принимать только значения, соответствующие элементам перечислимого типа. |

|  |  |
| --- | --- |
| **Поле:** | AlgorithmType |
| **Тип:** | SegmentationAlgorithmType |
| **Описание:** | Тип алгоритма, по которому построена сегментация |
| **Ограничения:** | Может принимать только значения, соответствующие элементам перечислимого типа. |

|  |  |
| --- | --- |
| **Поле:** | BioSdkVersion |
| **Тип:** | string |
| **Описание:** | Версия Sdk |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов. |

|  |  |
| --- | --- |
| **Поле:** | Quality |
| **Тип:** | double |
| **Описание:** | Качество |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | TimeSpeech |
| **Тип:** | double |
| **Описание:** | Длительность речи диктора |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | SnrIntegralDb |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | RtAver |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | DictorNumber |
| **Тип:** | int? |
| **Описание:** | Номер диктора |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | LowestSpeechFreqHz |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | HighestSpeechFreqHz |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | SpeechAmplitudeThresholdDb |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | RtRms |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | RtCount |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | TimeOverload |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | MaxUnclippedInterval |
| **Тип:** | double |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | MinDictorsCount |
| **Тип:** | int? |
| **Описание:** | Минимальное кол-во дикторов для алгоритма |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | MaxDictorsCount |
| **Тип:** | int? |
| **Описание:** | Максимальное кол-во дикторов для алгоритма |
| **Ограничения:** | - |

public interface ISegmentationMetaData

{

    ChannelNumber ChannelNumber { get; }

    ChannelType ChannelType { get; }

    SegmentationAlgorithmType AlgorithmType { get; }

    string BioSdkVersion { get; }

    double Quality { get; }

    double TimeSpeech { get; }

    double SnrIntegralDb { get; }

    double RtAver { get; }

    int? DictorNumber { get; }

    double LowestSpeechFreqHz { get; }

    double HighestSpeechFreqHz { get; }

    double SpeechAmplitudeThresholdDb { get; }

    double RtRms { get; }

    int RtCount { get; }

    double TimeTonalNoise { get; }

    double TimeOverload { get; }

    double MaxUnclippedInterval { get; }

    int? MinDictorsCount { get; }

    int? MaxDictorsCount { get; }

}

### ChannelNumber

| **Значение** | **Описание** |
| --- | --- |
| Left | Левый |
| Right | Правый |

В случае монофайла ChannelNumber будет Left.

public enum ChannelNumber

{

    Left,

    Right

}

### ChannelType

| **Значение** | **Описание** |
| --- | --- |
| Phone | Телефон |
| Microphone | Микрофон |

public enum ChannelType

{

    Phone,

    Microphone

}

### SegmentationAlgorithmType

| **Значение** | **Описание** |
| --- | --- |
| Vad | Детектирование чистой речи |
| Cpd | Рекомендуется для стереофайлов |
| Polylog | Рекомендуется для монофайлов |
| Manual | Ручная |

public enum SegmentationAlgorithmType

{

    Vad,

    Cpd,

    Polylog,

    Manual,

}

* 1. Лицевая информация

### Общая информация

В учетной системе для лица существует три понятия:

* файл − фотография;
* кроп лица − сегментация/область фотографии, по которой строится лицевая модель;
* лицевая модель − для учетной системы это ключ, по которому она хранится в ТСервисе, и сопроводительная метаинформация.

Лицевая модель ­− это по факту математическая абстракция; лицевые модели можно сравнивать (искать) между собой.

В систему учета могут попадать файлы без сегментаций и моделей.

В систему учета не должны попадать отдельно сегметации, по которым не построены модели, либо которые не привязаны к конкретному файлу, т. к., имея начальную фотографию, сегментации можно всегда построить через Т-Сервис.

В систему учета не должны попадать отдельно модели, т. е. модели, которые не связаны с конкретной сегментацией (сегментациями).

Т-Сервис умеет строить одну модель по нескольким сегментациям, но на практике смысла в этом нет, поэтому это не используется.

**Описание:**

Лицевая информация может содержать как просто фотофайлы, так и лицевую модель, построенную на основе одного из файлов.

На данный момент поддерживается лицевая модель построенная только из одного файла. На данный момент поддерживаются только фотографии (не видеофайлы).

### IFaceInfoData

|  |  |
| --- | --- |
| **Поле:** | ModelData |
| **Тип:** | IFaceModelData |
| **Описание:** | Содержит информацию о лицевой модели |
| **Ограничения:** | Должна быть **null**, если нет хотя бы одного файла с сегментацией. |

|  |  |
| --- | --- |
| **Поле:** | VideoFileWithSegmentationDatas |
| **Тип:** | IReadOnlyCollection<IVideoFileWithSegmentationData> |
| **Описание:** | Содержит информацию о фотофайлах |
| **Ограничения:** | * Не может содержать null в качестве элемента коллекции; * Не может равняться null, если IFaceInfoData != null; * Не может быть пустой коллекцией. |

public interface IFaceInfoData

{

    IReadOnlyCollection<IVideoFileWithSegmentationData> VideoFileWithSegmentationDatas { get; }

    IFaceModelData ModelData { get; }

}

### IFaceModelData

|  |  |
| --- | --- |
| **Поле:** | ModelKey |
| **Тип:** | string |
| **Описание:** | Ключ; на данный момент это путь хранения лицевой модели внутри TService |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов, если IFaceModelData != null. |

|  |  |
| --- | --- |
| **Поле:** | FaceModelMetaData |
| **Тип:** | IFaceModelMetaData |
| **Описание:** | Метаинформация по лицевой модели |
| **Ограничения:** | Не может принимать значение **null**, если IFaceModelData != null. |

public interface IFaceModelData

{

    string ModelKey { get; }

    IFaceModelMetaData FaceModelMetaData { get; }

}

### IFaceModelMetaData

|  |  |
| --- | --- |
| **Поле:** | FaceEngineMetaData |
| **Тип:** | IFaceEngineMetaData |
| **Описание:** | Метаинформация по движку |

|  |  |
| --- | --- |
| **Поле:** | Quality |
| **Тип:** | double |
| **Описание:** | Качество лицевой модели |

public interface IFaceModelMetaData

{

    IFaceEngineMetaData FaceEngineMetaData { get; }

    double Quality { get; }

}

### IFaceEngineMetaData

|  |  |
| --- | --- |
| **Поле:** | Profile |
| **Тип:** | string |
| **Описание:** | Профиль движка |

|  |  |
| --- | --- |
| **Поле:** | Version |
| **Тип:** | string |
| **Описание:** | - |

|  |  |
| --- | --- |
| **Поле:** | ModelVersion |
| **Тип:** | string |
| **Описание:** | - |

public interface IFaceEngineMetaData

{

    string Profile { get; }

    string Version { get; }

    string ModelVersion { get; }

}

### IVideoFileWithSegmentationData

|  |  |
| --- | --- |
| **Поле:** | FileKey |
| **Тип:** | string |
| **Описание:** | Ключ файла; на данный момент это ключ хранения файла на Storage |
| **Ограничения:** | 1. Не может принимать значение **null**, **empty** или состоять только из пробельных символов. Уникально внутри коллекции VideoFileWithSegmentationDatas. |

|  |  |
| --- | --- |
| **Поле:** | FileMetaData |
| **Тип:** | IFileMetaData |
| **Описание:** | Фото (видео) метаинформация |
| **Ограничения:** | Не может принимать значение null. |

|  |  |
| --- | --- |
| **Поле:** | VideoMetaData |
| **Тип:** | IVideoMetaData |
| **Описание:** | Файловая метаинформация |
| **Ограничения:** | Не может принимать значение **null**. |

|  |  |
| --- | --- |
| **Поле:** | SegmentationData |
| **Тип:** | IFaceSegmentationData |
| **Описание:** | Информация о сегментации |
| **Ограничения:** | 1. - |

На данный момент только у одного IVideoFileWithSegmentationData будет заполнен SegmentationData −тот, по которому построена модель.

### IFileMetaData

|  |  |
| --- | --- |
| **Поле:** | AddedAt |
| **Тип:** | DateTimeOffset |
| **Описание:** | Время добавление файла в UTC |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | CreatedByUserIdentifier |
| **Тип:** | object |
| **Описание:** | Идентификатор пользователя, добавившего файл. На данный момент это Guid − учетный идентификатор пользователя |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | FileName |
| **Тип:** | string |
| **Описание:** | - |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов. |

|  |  |
| --- | --- |
| **Поле:** | FileExtension |
| **Тип:** | string |
| **Описание:** | - |
| **Ограничения:** | - |

|  |  |
| --- | --- |
| **Поле:** | BinHash |
| **Тип:** | string |
| **Описание:** | Бинарный хэш файла |
| **Ограничения:** | Не может принимать значение **null**, **empty** или состоять только из пробельных символов. |

public interface IFileMetaData

{

    DateTimeOffset AddedAt { get; }

    object CreatedByUserIdentifier { get; }

    string FileName { get; }

    string FileExtension { get; }

    string BinHash { get; }

}

### IVideoMetaData

|  |  |
| --- | --- |
| **Поле:** | Comment |
| **Тип:** | string |
| **Описание:** | Комментарий к файлу. |

На данный момент пока не заполняется, поскольку в UI нет сценариев с заполнением.

public interface IVideoMetaData

{

    string Comment { get; }

}

### IFaceSegmentationData

|  |  |
| --- | --- |
| **Поле:** | SegmentationKey |
| **Тип:** | string |
| **Описание:** | Ключ; на данный момент это путь хранения голосовой сегментации по файлу внутри TService |
| **Ограничения:** | 1. Не может принимать значение **null**, **empty** или состоять только из пробельных символов. Уникально внутри коллекции VideoFileWithSegmentationDatas. |

|  |  |
| --- | --- |
| **Поле:** | SegmentationMetaData |
| **Тип:** | IFaceSegmentationMetaData |
| **Описание:** | Метаинформация по сегментации по файлу |
| **Ограничения:** | Не может принимать значение null. |

public interface IFaceSegmentationData

{

    string SegmentationKey { get; }

    IFaceSegmentationMetaData SegmentationMetaData { get; }

}

### IFaceSegmentationMetaData

|  |  |
| --- | --- |
| **Поле:** | X |
| **Тип:** | int |
| **Описание:** | X координата верхнего левого угла кропа |

|  |  |
| --- | --- |
| **Поле:** | Y |
| **Тип:** | int |
| **Описание:** | Y координата верхнего левого угла кропа |

|  |  |
| --- | --- |
| **Поле:** | Width |
| **Тип:** | int |
| **Описание:** | Ширина кропа |

|  |  |
| --- | --- |
| **Поле:** | Height |
| **Тип:** | int |
| **Описание:** | Высота кропа |

|  |  |
| --- | --- |
| **Поле:** | EngineMetaData |
| **Тип:** | IFaceEngineMetaData |
| **Описание:** | Метаинформация по движку |

|  |  |
| --- | --- |
| **Поле:** | EyeDistance |
| **Тип:** | double? |
| **Описание:** | Расстояние между глазами |

|  |  |
| --- | --- |
| **Поле:** | PointsCount |
| **Тип:** | double? |
| **Описание:** | Количество опорных точек |

|  |  |
| --- | --- |
| **Поле:** | EstimatedDeviation |
| **Тип:** | IEstimatedDeviation |
| **Описание:** | - |

|  |  |
| --- | --- |
| **Поле:** | PortraitCharacteristics |
| **Тип:** | IPortraitCharacteristics |
| **Описание:** | Портретные характеристики |

public interface IFaceSegmentationMetaData : ICropArea

{

    IFaceEngineMetaData EngineMetaData { get; }

    double? EyeDistance { get; }

    double? PointsCount { get; }

    IEstimatedDeviation EstimatedDeviation { get; }

    IPortraitCharacteristics PortraitCharacteristics { get; }

}

public interface ICropArea

{

    int X { get; }

    int Y { get; }

    int Height { get; }

    int Width { get; }

}

### IEstimatedDeviation

|  |  |
| --- | --- |
| **Поле:** | Yaw |
| **Тип:** | double? |
| **Описание:** | - |

|  |  |
| --- | --- |
| **Поле:** | Roll |
| **Тип:** | double? |
| **Описание:** | - |

|  |  |
| --- | --- |
| **Поле:** | Pitch |
| **Тип:** | double? |
| **Описание:** | - |

public interface IEstimatedDeviation

{

    double? Yaw { get; }

    double? Roll { get; }

    double? Pitch { get; }

}

### IPortraitCharacteristics

|  |  |
| --- | --- |
| **Поле** | **Тип** |
| NoDarkGlasses | IDecisionScore |
| NoMustache | IDecisionScore |
| MouthClosed | IDecisionScore |
| LeftEyeOpen | IDecisionScore |
| RightEyeOpen | IDecisionScore |

public interface IPortraitCharacteristics

{

    double? GrayScaleDensity { get; }

    double? FrontalPoseDeviation { get; }

    double? DeviationFromUniformLighting { get; }

    double? HotSpots { get; }

    double? Exposure { get; }

    double? Sharpness { get; }

    IDecisionScore Male { get; }

    IDecisionScore Female { get; }

    IDecisionScore NoBeard { get; }

    IDecisionScore NoSmile { get; }

    IDecisionScore NoGlasses { get; }

    IDecisionScore NoDarkGlasses { get; }

    IDecisionScore NoMustache { get; }

    IDecisionScore MouthClosed { get; }

    IDecisionScore LeftEyeOpen { get; }

    IDecisionScore RightEyeOpen { get; }

}

public interface IDecisionScore

{

    double? Score { get; }

    bool? Decision { get; }

}

1. StorageModule

**(модуль хранения бинарных данных)**

* 1. Тип 1

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/upload |
| **Описание:** | Запись данных в хранилище. Возвращает структуру со следующими полями:   * DateTimeOffset StorageKeyCreationDate – дата создания ключа storageKey; * DateTimeOffset UploadDate – дата первой загрузки данных; * string FileMetadata – метаданные загруженных данных. При использовании этого метода всегда **null**; * string StorageKey. |
| **Пример ответа:** | {  "StorageKey": "string",  "StorageKeyCreationDate": "2020-10-27T11:12:21.611Z",  "UploadDate": "2020-10-27T11:12:21.611Z",  "FileMetadata": null  } |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | - |
| **Примечание:** | * Если длина данных равна 0, вернется ошибка 400. Поток передаётся посредством тела запроса. |

* 1. Тип 2

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/upload-with-meta |
| **Описание:** | Запись данных с метаданными в хранилище. Возвращает структуру со следующими полями:   * DateTimeOffset StorageKeyCreationDate – дата создания ключа storageKey; * DateTimeOffset UploadDate – дата первой загрузки данных; * string FileMetadata – метаданные загруженных данных. При использовании этого метода всегда отлично от **null**; * string StorageKey. |
| **Пример ответа:** | {    "StorageKey": "string",    "StorageKeyCreationDate": "2020-10-27T11:12:21.611Z",    "UploadDate": "2020-10-27T11:12:21.611Z",    "FileMetadata": "some metadata at string format"  } |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | - |
| **Примечание:** | * Если длина данных равна 0, вернется ошибка 400. Ошибка 400 если метаданные: * равны пустой строке; * если состоят только из пробелов; * имеют длину больше 1024. * Данные в запросе передаются в виде "multipart/form-data" контента. "multipart/form-data" контент состоит из: * файлового потока; * метаданных в виде одной строки, доступных по ключу "metadata". * Если по ключу "metadata" в контенте найдено несколько строк, то возвращается ошибка 400. |

* 1. Тип 3

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/append/{storageKey} |
| **Описание:** | Дописывание данных. Рекомендуемый лимит данных 56 kB. |
| **Пример ответа:** | Cтандартный 200 или ошибка. |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | * При длине дописываемых данных, равной нулю, результат работы всегда успешный, даже если данных с таким storageKey нет; * Имеющиеся метаданные файла по storageKey не меняются; * Массив байт размещается в теле запроса. |

* 1. Тип 4

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/append-stream/{storageKey} |
| **Описание:** | Дописывание данных. Рекомендуемый лимит данных 56 kB. |
| **Пример ответа:** | Cтандартный 200 или ошибка. |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | * При длине дописываемых данных, равной нулю, результат работы всегда успешный, даже если данных с таким storageKey нет; * Имеющиеся метаданные файла по storageKey не меняются; * Поток передаётся посредством тела запроса. |

* 1. Тип 5

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/append-stream-with-meta/{storageKey} |
| **Описание:** | Дописывание данных с перезаписью метаданных. Рекомендуемый лимит данных 56 kB. |
| **Пример ответа:** | Cтандартный 200 или ошибка. |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | Ошибка 400, если метаданные:   * равны пустой строке; * состоят только из пробелов; * имеют длину больше 1024.   Старые метаданные файла для указанного storageKey полностью заменяются новыми. Данные в запросе передаются в виде "multipart/form-data" контента. "multipart/form-data" контент состоит из:   * файлового потока; * метаданных в виде одной строки, доступных по ключу "metadata".   Если по ключу "metadata" в контенте найдено несколько строк, то возвращается ошибка 400. |

* 1. Тип 6

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/rewrite/{storageKey} |
| **Описание:** | Загружает файл и перепривязывает к нему storageKey. |
| **Пример ответа:** | (аналогично upload):  {  "StorageKey": "string",  "StorageKeyCreationDate": "2020-10-27T11:12:21.611Z",  "UploadDate": "2020-10-27T11:12:21.611Z",  "FileMetadata": null  } |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | * Ошибка 404, если данных с таким storageKey нет; * Ошибка 400, если длина файла равна 0; * Имеющиеся метаданные файла по storageKey не меняются; * Поток передаётся посредством тела запроса. |

* 1. Тип 7

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/rewrite-with-meta/{storageKey} |
| **Описание:** | Загружает файл и перепривязывает к нему storageKey, перезаписывая метаданные. |
| **Пример ответа:** | (аналогично upload):  {  "StorageKey": "string",  "StorageKeyCreationDate": "2020-10-27T11:12:21.611Z",  "UploadDate": "2020-10-27T11:12:21.611Z",  "FileMetadata": null  } |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | * Ошибка 404, если данных с таким storageKey нет; * Ошибка 400, если длина файла равна 0; * Ошибка 400 если метаданные: * равны пустой строке; * состоят только из пробелов; * имеют длину больше 1024.   Старые метаданные файла для указанного storageKey полностью заменяются новыми. Данные в запросе передаются в виде "multipart/form-data" контента. "multipart/form-data" контент состоит из:   * файлового потока; * метаданных в виде одной строки, доступных по ключу "metadata".   Если по ключу "metadata" в контенте найдено несколько строк, то возвращается ошибка 400. |

* 1. Тип 8

|  |  |
| --- | --- |
| **Тип:** | GET |
| **Путь:** | api/v2/storage/download/{storageKey} |
| **Описание:** | Скачивание данных по ключу |
| **Пример ответа:** | - |
| **Необходимые Permissions:** | StoragePermissions.StorageRead |
| **DLock\*:** | - |
| **Примечание:** | - |

* 1. Тип 9

|  |  |
| --- | --- |
| **Тип:** | GET |
| **Путь:** | api/v2/storage/metadata/{storageKey} |
| **Описание:** | Получение всей метаинформации по ключу; структура аналогична п. 1 или п. 2. |
| **Пример ответа:** | Если метаданные файла не были загружены (аналогично upload):  {  "StorageKey": "string",  "StorageKeyCreationDate": "2020-10-27T11:12:21.611Z",  "UploadDate": "2020-10-27T11:12:21.611Z",  "FileMetadata": null  }  Если метаданные файла были загружены (аналогично upload-with-meta):  {  "StorageKey": "string",  "StorageKeyCreationDate": "2020-10-27T11:12:21.611Z",  "UploadDate": "2020-10-27T11:12:21.611Z",  "FileMetadata": "some metadata at string format"  } |
| **Необходимые Permissions:** | StoragePermissions.StorageRead |
| **DLock\*:** | - |
| **Примечание:** | Метаданные файла равны null, если не были загружены. |

* 1. Тип 10

|  |  |
| --- | --- |
| **Тип:** | POST |
| **Путь:** | api/v2/storage/create-additional-storage-key/{storageKey} |
| **Описание:** | Создаёт дополнительный storageKey для данных, относящихся к переданному storageKey. Возвращает структуру со следующими полями:   * DateTimeOffset StorageKeyCreationDate – дата создания дополнительного storageKey; * DateTimeOffset UploadDate - дата первой загрузки бинарных данных, к которым привязаны переданный и созданный storageKey; * string FileMetadata – метаданные нового storageKey, которые были скопированы из переданного storageKey; * string StorageKey – созданный новый storageKey. |
| **Пример ответа:** | {  "StorageKey": "string",  "StorageKeyCreationDate": "2021-05-03T11:12:21.611Z",  "UploadDate": "2020-10-27T11:12:21.611Z",  "FileMetadata": "some string metadata"  } |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | * Ошибка 400: * при равенстве null; * при равенстве пустой строке; * если содержит символ пробела; * если состоит только из символов '/'; * если длина больше 512. * Ошибка 404, если данных с таким storageKey нет: * Новый и переданный storageKey ничего не знают друг о друге.   Если после создания по переданному старому storageKey изменяются данные (методы Append, Rewrite, Delete) и (или) метаданные, то для нового storageKey данные и (или) метаданные остаются прежними. |

* 1. Тип 11

|  |  |
| --- | --- |
| **Тип:** | DELETE |
| **Путь:** | api/v2/storage/delete/{storageKey} |
| **Описание:** | Отвязывает ключ от файла. Если больше нет ключей, связанных с файлом, файл помечается на удаление, и, спустя некоторое время, он будут удален из хранилища. |
| **Пример ответа:** | - |
| **Необходимые Permissions:** | StoragePermissions.StorageWrite |
| **DLock\*:** | 1. storageKey;  2. fileDescriptor.Id |
| **Примечание:** | - |

* 1. Тип 12

|  |  |
| --- | --- |
| **Тип:** | GET |
| **Путь:** | api/v2/storage/health-check |
| **Описание:** | Проверка heartbeat |
| **Пример ответа:** | - |
| **Необходимые Permissions:** | - |
| **DLock\*:** | - |
| **Примечание:** | - |

* 1. Тип 13

|  |  |
| --- | --- |
| **Тип:** | GET |
| **Путь:** | api/v2/storage/NginxStream/GetNginxStream/{id} |
| **Описание:** | Использует Nginx (редирект на путь к файлу) |
| **Пример ответа:** | - |
| **Необходимые Permissions:** | StoragePermissions.StorageRead |
| **DLock\*:** | - |
| **Примечание:** | - |

* 1. Тип 14

|  |  |
| --- | --- |
| **Тип:** | HEAD |
| **Путь:** | api/v2/storage/NginxStream/GetNginxStream/{id} |
| **Описание:** | Использует Nginx (редирект на путь к файлу) |
| **Пример ответа:** | - |
| **Необходимые Permissions:** | StoragePermissions.StorageRead |
| **DLock\*:** | - |
| **Примечание:** | - |

001\_17052022



**Санкт-Петербург**

Выборгская наб., 45Е, 194044

+7 812 325 8848

stc-int@speechpro.com

**Москва**

Земляной Вал, 59/2, 109004

+7 495 669 7440

stc-int@speechpro.com