УТВЕРЖДЕН

DOC-XX.XXX.XXX.XXX -ЛУ

Документ содержит **описание API подсистемы «Диалоговый ассистент»** — веб-интерфейса, созданного для поддержки пользователей госуслуг. Приведены методы, параметры, форматы запросов и ответов. Спецификация предназначена для разработчиков и интеграторов и помогает упростить подключение и обеспечить масштабируемость. Система улучшает доступ к сервисам и снижает нагрузку на операторов.

ПОДСИСТЕМА «ОНЛАЙН-КОНСУЛЬТАНТ»

ФЕДЕРАЛЬНОЙ ГОСУДАРСТВЕННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Описание программного интерфейса «Service API»

DOC-XX.XXX.XXX.XXX

Государственный контракт от 32 мартобря 20XX г. № XX.XXXX.XXX

Листов 30

Москва-20XX
https://dev.portal.local/

Литера **У**

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

АННОТАЦИЯ

Настоящий документ содержит Описание программного интерфейса «Service API[1]» Подсистемы «Онлайн-консультант» Федеральной Государственной Информационной Системы. (далее по тексту ОК[2]).

Документ оформлен в соответствии с правилами, предусмотренными ГОСТ Р 59795–2021, ГОСТ 2.301-68, ГОСТ 34.201-2020, ГОСТ 19.105, ГОСТ 19.106 и другими стандартами Единой системы программной документации (ЕСПД).

[1] Service API (SAPI) v1.0
[2] ОК – Онлайн консультант

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

В настоящем документе содержится описание программного интерфейса «OK Service API» (далее – SAPI), предназначенного для подключения в качестве шлюза обмена текстовыми сообщениями к сервису диалоговых ассистентов OK.

SAPI представляет собой асинхронный программный интерфейс подключения клиентского приложения или сервиса (далее - клиент) к OK для обмена текстовыми сообщениями с ассистентом.

2 ТРЕБОВАНИЯ ПОДКЛЮЧЕНИЯ

Для использования интерфейса SAPI необходимо обладать следующими данными:

– DSN интерфейса SAPI (для публичного облака: https://dev.portal.local/), далее – SAPI-DSN.

- Данные авторизации.

– Идентификатор канала (ChannelUID) ассистента в ОК к которому производится подключение (его можно получить в интерфейсе ОК путем создания канала с типом Server API). Данный канал должен быть активирован.

2.1 Базовые понятия и определения

Ниже приведен краткий список определений, используемых при работе с ОК и SAPI.

- Собеседник – конечный пользователь системы, ведущий диалог с ассистентом.
- Диалог – обмен сообщениями между конкретным Собеседником и ассистентом. Диалог разбивается на последовательные Сессии разделенные периодом отсутствия активности Собеседника.
- ChannelUID – уникальный идентификатор (UUID) канала ассистента в ОК (его можно создать через DOS Logic UI).
- DialogUID – уникальный идентификатор (UUID) диалога Собеседника с ассистентом.
- SAPI-DSN - DSN интерфейса SAPI.

3 АВТОРИЗАЦИЯ

Авторизация пока отключена, в ближайшее время она будет включена.

Процесс авторизации в API обычно включает в себя несколько этапов:

- Запрос: Пользователь отправляет запрос на авторизацию, содержащий необходимые данные (например, логин и пароль).
- Проверка подлинности: API проверяет, соответствуют ли предоставленные пользователем данные (например, логин и пароль) данным, хранящимся в системе. Если данные совпадают, пользователь считается аутентифицированным.
- Генерация токена: Если пользователь успешно прошел проверку подлинности, API генерирует токен, который представляет собой уникальный идентификатор, связанный с пользователем. Токен может использоваться для идентификации пользователя при последующих запросах к API.
- Возврат токена: API возвращает токен пользователю, который может использовать его для авторизации при последующих запросах.
- Хранение токена: Пользователь должен сохранить полученный токен для использования в будущих запросах. Обычно токены имеют ограниченный срок действия, после истечения которого необходимо получить новый токен.

Этот процесс может отличаться в зависимости от конкретного API и требований к безопасности. Некоторые API могут использовать дополнительные методы проверки подлинности, такие как двухфакторная аутентификация.

3.1 Инициализация клиента

Перед подключением Клиента к OK через интерфейс SAPI (Рисунок 1) необходимо установить webhook на который по протоколу HTTPs будут отправляться все сообщения от OK. Установка webhook'a производится с помощью вызова функции /setWebhook, в результате которого для указанного канала целевого ассистента OK будет зарегистрирован указанный DSN для передачи сообщений от OK.

Для проверки ранее установленных параметров webhook можно использовать запрос /getWebhook.

Рисунок 1 – Общая схема использования интерфейса SAPI

3.2 Ведение диалога с собеседником

Диалог с ассистентом (Рисунок 2) начинается с инициализации или реинициализации (если он является продолжением диалога, произошедшего ранее). Инициализация производится с помощью вызова метода /startDialog, в результате которого возвращается уникальный идентификатор диалога.

После инициализации диалога имеет смысл запросить приветственное сообщение. Приветственное сообщение возвращается в ответ на событие начала диалога.

Дальнейший диалог (Рисунок 2) состоит из асинхронного обмена запросами между клиентом и SAPI в асинхронном режиме и состоит из следующих возможных запросов:

1) Передача текстового сообщения от Собеседника: Выполняется через вызов метода /sendRequest, который позволяет отправить текстовое сообщение ассистенту в рамках текущего диалога.

2) Передача события диалога: Выполняется через вызов метода /sendEvent, который позволяет отправить событие ассистенту в рамках текущего диалога.

3) Оценка ранее полученного ответа ассистента: Выполняется через вызов метода /rateResoponse, который позволяет оценить ранее полученный ответ от ассистента в рамках текущего диалога.

4) Получение ответа от ассистента: Сообщение от ассистента отправляются на webhook, установленный через вызов метода /setWebhook или переданный в качестве параметра reply_to методов /sendRequest, /sendEvent и /rateResoponse в описанном формате.

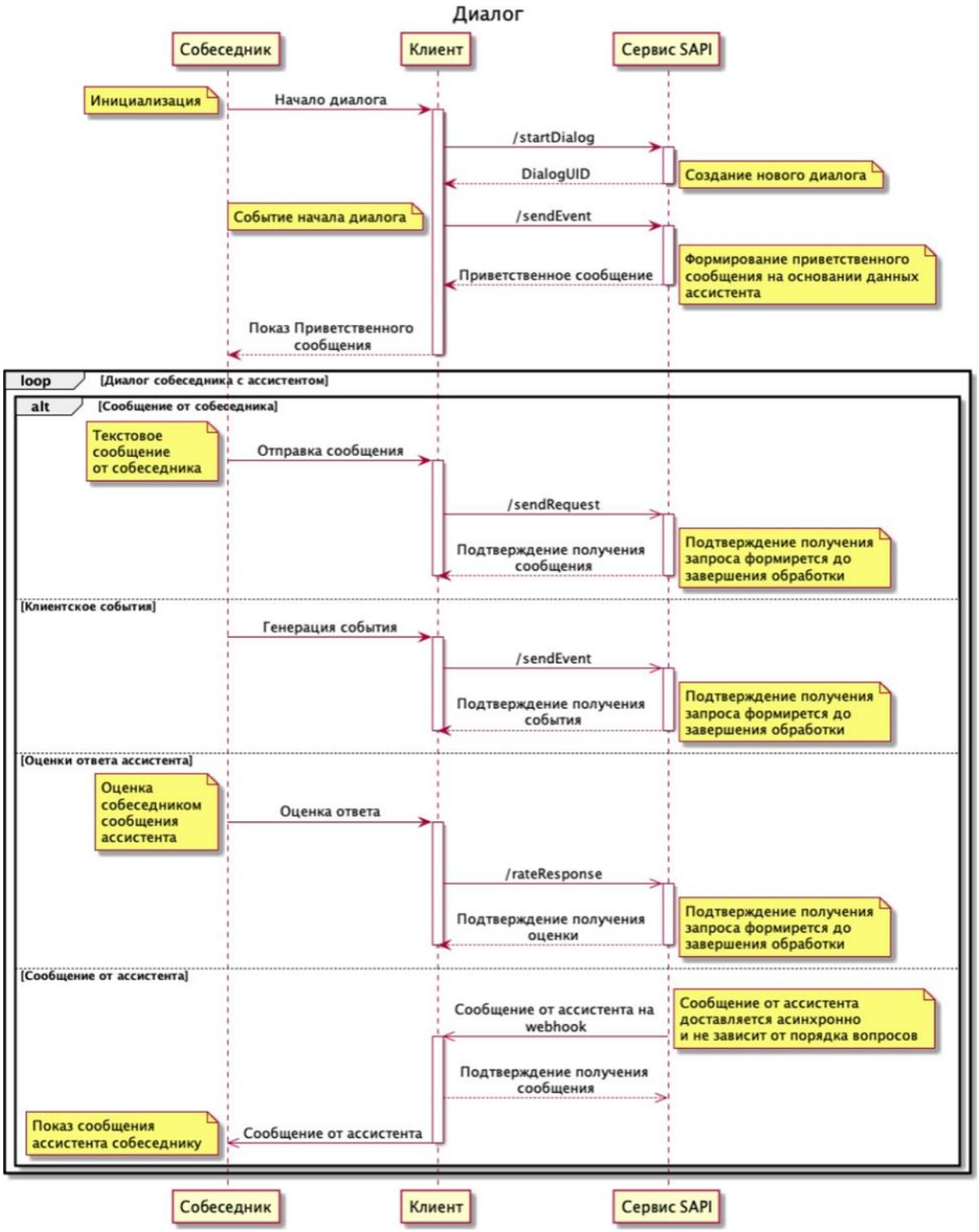

Рисунок 2 – Ведение диалога с собеседником

4 ОБЩИЕ СТРУКТУРЫ ДАННЫХ

Ниже описаны специальные структуры данных, используемые при работе с SAPI.

Передача файлов (в настоящее время в виде url)

- file - `{type: json}` - вложение файлов
- ref - `{type: url}` - ссылка

1) filename - `{type: string}` - имя файла, которое будет предложено при скачивании

Данные DSN для отправки сообщений от ассистента (reply_to)

- reply_to - `{type: json}`:

1) host – {type: string}– адрес (DSN) для отправки сообщений от ассистента.

2) auth_key – {type: string}– ключ, передаваемый в заголовке X-NLab-WebHook-Key вместе с каждым сообщением.

4.1.1 Контекст (context)

Контекст представляет собой JSON структуру, содержащую набор переменных. Имена и возможные значения используемых переменных строго специфицированы для каждого канала и метода API, не специфицированные переменные игнорируются Сервисом.

Переменные контекста используются для передачи информации, необходимой для формирования ответов на запросы к ассистенту и поддержания общего диалога.

Сообщение ассистента (response)

Сообщение ассистента представляет собой массив элементов, описанных ниже:

а) текст

Элемент предназначен для отображения текстовых данных.

Элемент представляет собой JSON структуру со следующими полями:

- type – {type: string}: text – фиксированное значение типа элемента.
- text - {type: string} – текстовые данные для отображения.

б) кликабельный ответ

Элемент предназначен для отображения кликабельного текстового элемента, клик по которому приводит к автоматической отправке сообщения ассистенту.

Элемент представляет собой JSON структуру со следующими полями:

- type – {type: string}: userlink – фиксированное значение типа элемента.
- ink – {type: string} – текстовые данные для отображения.
- request – {type: string, optional: true} – запрос, отправляемый ассистенту при клике по элементу. Если это поле отсутствует или не заполнено, то вместо него используется значение поля link.

в) ссылка

Элемент предназначен для отображения кликабельной ссылки.

Элемент представляет собой JSON структуру со следующими полями:

- type – {type: string}: link – фиксированное значение типа элемента.
- link - {type: string}– текст для отображения. Если это поле отсутствует, то вместо него используется полный адрес ссылки.
- ref - {type: string}- адрес документа (URL, Universal Resource Locator, универсальный указатель ресурсов), на который следует перейти при клике по элементу.
- target - {type: string}- имя окна или фрейма, куда браузер будет загружать документ.

г) перенос строки

Элемент предназначен для начала новой строки в рамках ответа ассистента.

Элемент представляет собой JSON структуру со следующими полями:

- type – {type: string}: br – фиксированное значение типа элемента.

д) изображение

Элемент предназначен для отображения изображений в графическом формате gif, jpeg или png. Адрес файла с картинкой задаётся через атрибут src.

Элемент представляет собой JSON структуру со следующими полями:

- type – {type: string}: img – фиксированное значение типа элемента.
- src – {type: string} – путь к графическому файлу.
- alt – {type: string, optional: true}– альтернативный текст для изображения.

е) список

Элемент предназначен для отображения набора маркированного списка.

Элемент представляет собой JSON структуру со следующими полями:

- type – {type: string}: list – фиксированное значение типа элемента.

1) ordered – {type: bool, optional: true} – признак нумерованного списка. По умолчанию: true.

2) items – {type: array}– массив элементов списка, где каждый элемент представляет собой JSON структуру со следующими полями:

3) type – {type: string}: item – фиксированное значение типа элемента.

4) values - {type: array[response]} – массив элементов списка, представленных объектами типа response.

5 МЕТОДЫ СЕРВИСА

5.1 Общие параметры API

5.1.1 Протокол

В качестве транспорта для взаимодействия с SAPI используется HTTPS с TLS шифрованием (HTTP не поддерживается). Данные передаются в формате JSON в кодировке UTF-8.

5.1.2 Headers

При всех обращениях к SAPI необходимо использовать следующие HTTP-заголовки:

- Content-Type: application/json

5.2 Установка WebHook'а (/setWebhook)

Настройка адреса для отправки сообщений передаваемых OK.

При отправке любого запроса на webhook устанавливается заголовок X-NLab-WebHook-Key со значением key, который позволят идентифицировать источник оповещения на принимающей стороне.

В процессе обработки запроса производится HTTP запрос к указанному webhook'у, в ответ на который ожидается ответ с HTTP статус кодом 200 и телом, совпадающим со значением параметра verify. Если сервис ответит другим кодом или иным телом ответа, то проверка будет считаться неуспешной.

5.2.1.1 Запрос

a) Method

- POST: {SAPI-DSN}/api/v1/setWebhook/{ChannelUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, для которого устанавливается webhook.

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

- url – {type: string} – адрес (DSN) для отправки сообщений от ассистента.
- key – {type: string} – ключ, передаваемый в заголовке X-NLab-WebHook-Key вместе с каждым сообщением.
- verify – {type: string}– строка, которую сервис должен возвращать в качестве body при проверке webhook's.

5.2.1.2 Результаты

а) Success

- HTTP Code: 200
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: true – фиксированное значение.

б) Error

- HTTP Code: 4** / 500
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}:false – фиксированное значение.
        б) result – {type: object}:
        в) error_type – {type: string} – тип ошибки.
        г) error_message – {type: string}– описание ошибки.
        д) Проверка WebHook'a (/getWebhook).

Проверка установленного webhook, на который по протоколу HTTPs будут отправляться все сообщения от OK.

5.2.1.3 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/getWebhook/{ChannelUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, для которого запрашиваются данные webhook's.

5.2.1.4 Результаты

а) Успех

- HTTP Code: 200
  1) Тело ответа представляет собой JSON структуру со следующими полями:
     а) success – {type: bool}:true – фиксированное значение.
     б) url – {type: string}– адрес (DSN) для отправки сообщений от ассистента.
     в) key – {type: string}– ключ, передаваемый в заголовке X-NLab-WebHook-Key вместе с каждым сообщением.
     г) verify – {type: string}– строка, которую сервис должен возвращать в качестве body при проверке webhook's.

б) Ошибка

- HTTP Code: 4** / 500
  1) Тело ответа представляет собой JSON структуру со следующими полями:
     а) success – {type: bool}: false – фиксированное значение.
     б) result – {type: object}:
     в) error_type – {type: string} – тип ошибки.
     г) error_message – {type: string}– описание ошибки.
     д) Инициализация/Реинициализация диалога (/startDialog)

е) Инициализация или реинициализация (повторная инициализация существующего диалога).

5.2.1.5 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/startDialog/{ChannelUID}
- POST: {SAPI-DSN}/api/v1/startDialog/{ChannelUID}/{DialogUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, в рамках которого происходит инициализация диалога.
- DialogUID – {type: UUID, optional: true} – идентификатор текущего диалога для реинициализации.

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

1) context – {type: json}– настройка начального контекста диалога.
2) extra * - {type: json}- дополнительные параметры. deprecated
3) external_user_id * - {type: string}- id внешнего пользователя чата (перемещено в ** extra **)
4) extarnal_chat_id * - {type: string}- id внешнего чата (перемещено в ** extra **)
5) extra - {type: json}- дополнительные (технические) параметры чата
6) stealth * - {type: boolean} - признак скрытого диалога
7) external_user_id * - {type: string}- id внешнего пользователя чата
8) extarnal_chat_id * - {type: string}- id внешнего чата
9) new – {type: bool, optional: true} – флаг необходимости начать новую сессию диалога, даже если диалог активен.

5.2.1.6 Результаты

а) Успех

- HTTP Code: 200
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: true – фиксированное значение.
        б) dialog_uid – {type: UUID}– уникальный идентификатор диалога.

б) Ошибка

- HTTP Code: 4** / 500
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: false – фиксированное значение.
        б) result – {type: object}:
        в) error_type – {type: string}– тип ошибки.
        г) error_message – {type: string}– описание ошибки.

5.3 Отправка сообщения (/sendRequest)

Отправка сообщения ассистенту.

5.3.1.1 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/sendRequest/{ChannelUID}/{DialogUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, в рамках которого происходит диалог.
- DialogUID – {type: UUID}– идентификатор диалога.

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

- message – {type: string}– текст сообщения ассистенту.
- context – {type: context, optional: true}- контекст сообщения.
- reply_to – {type: reply_to, optional: true} –адрес (DSN) для отправки ответных сообщений от ассистента для конкретного сообщения.

1) files - {type: files}

5.3.1.2 Результаты

а) Успех

- HTTP Code: 200
  1) Тело ответа представляет собой JSON структуру со следующими полями:
     а) success – {type: bool}: true – фиксированное значение.
     б) reqid – {type: UUID}– уникальный идентификатор сообщения.

б) Ошибка

- HTTP Code: 4** / 500
  1) Тело ответа представляет собой JSON структуру со следующими полями:
     а) success – {type: bool}: false – фиксированное значение.
     б) result – {type: object}:
     в) error_type – {type: string}– тип ошибки.
     г) error_message – {type: string}– описание ошибки.

5.4 Отправка события (/sendEvent)

Отправка события ассистенту.

5.4.1.1 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/sendEvent/{ChannelUID}/{DialogUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, в рамках которого происходит диалог.
- DialogUID – {type: UUID}– идентификатор диалога.

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

- event_uid – {type: UUID}– идентификатор события.
- context – {type: context, optional: true}- контекст сообщения.
- reply_to – {type: reply_to, optional: true} –адрес (DSN) для отправки ответных сообщений от ассистента для конкретного сообщения.
    1) files - {type: array[file]}

5.4.1.2 Результаты

а) Успех

- HTTP Code: 200
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: true – фиксированное значение.
        б) reqid – {type: UUID}– уникальный идентификатор события.

б) Ошибка

- HTTP Code: 4** / 500
    1) Тело ответа представляет собой JSON структуру со следующими полями:

а) success – {type: bool}: false – фиксированное значение.

б) result – {type: object}:

в) error_type – {type: string}– тип ошибки.

г) error_message – {type: string}– описание ошибки.

5.5 Отправка события (/event). Deprecated

Отправка события ассистенту.

Метод /event больше не рекомендуется использовать.

5.5.1.1 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/event/{ChannelUID}/{DialogUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, в рамках которого происходит диалог.
- DialogUID – {type: UUID}– идентификатор диалога.

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

- event_uid – {type: UUID}– идентификатор события.
- context – {type: context, optional: true} - контекст сообщения.
- reply_to – {type: reply_to, optional: true}– адрес (DSN) для отправки ответных сообщений от ассистента для конкретного сообщения.

1) files - {type: array[file]}

5.5.1.2 Результаты

а) Успех

- HTTP Code: 200
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: true – фиксированное значение.
        б) reqid – {type: UUID}– уникальный идентификатор события.

б) Ошибка

- HTTP Code: 4** / 500
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: false – фиксированное значение.
        б) result – {type: object}:
        в) error_type – {type: string}– тип ошибки.
        г) error_message – {type: string}– описание ошибки.

5.6 Оценка ответа ассистента (/rateResponse)

Оценка полученного от ассистента ответа.

5.6.1.1 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/rateResponse/{ChannelUID}/{DialogUID}

б) Arguments

- ChannelUID – {type: UUID} – идентификатор канала ассистента, в рамках которого происходит диалог.
- DialogUID – {type: UUID}– идентификатор диалога.

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

- reqid – {type: UUID}– идентификатор сообщения или события.
  1) context – {type: context, optional: true}- контекст сообщения.
  2) reply_to – {type: reply_to, optional: true}– адрес (DSN) для отправки ответных сообщений от ассистента для конкретной оценки.
  3) rate – {type: int, range: [0,9]}– оценка ответа ассистента.
  4) comment – {type: string}– комментарий к оценке.

5.6.1.2 Результаты

а) Успех

- HTTP Code: 200
  1) Тело ответа представляет собой JSON структуру со следующими полями:
     а) success – {type: bool}: true – фиксированное значение.
     б) reqid – {type: UUID}– уникальный идентификатор оценки.

б) Ошибка

- HTTP Code: 4** / 500
  1) Тело ответа представляет собой JSON структуру со следующими полями:
     а) success – {type: bool}: false – фиксированное значение.
     б) result – {type: object}:
     в) error_type – {type: string} – тип ошибки.
     г) error_message – {type: string}– описание ошибки.

5.7 Ответ Business Logic Service (/bls/response)

Ответ, полученный от ассистента.

/bls/response является внутренним методом, который не рекомендуется использовать во избежание негативных последствий.

5.7.1.1 Запрос

а) Method

- POST: {SAPI-DSN}/api/v1/bls/response

б) Headers

- kind – {type: string}- Тип сообщения в очереди (kind: request - ответ на запрос, kind:
- session.close - уведомление о закрытии сессии, kind: session.switch.assistant - уведомление о переходе сессии с ассистента на оператора)
- trace_id – {type: string}- Идентификатор запроса в Jaeger
- client_auth_key – {type: string, optional: true} – Клиентский ключ
    1) client_host – {type: string}– Адрес клиента
- reqid – {type: UUID}– Идентификатор запроса
- content_type – {type: string, optional: true}– applicatinon/json
- destination – {type: string, optional: true}- Целевой адресат в RabbitMQ
- duid – {type: UUID}– Идентификатор диалога
    1) buid – {type: UUID}– Идентификатор канала
    2) suid – {type: UUID}– Идентификатор сессии
- euid – {type: UUID}– Идентификатор события

в) Request Body

Тело запроса представляет собой JSON структуру со следующими полями:

- context – {type: object} - переменные контекста используются для передачи информации, необходимой для формирования ответов на запросы к ассистенту и поддержания общего диалога
- text – {type: object, optional: true}- элемент обозначает текстовое сообщение
    1) value – {type: string, optional: true}- текст сообщения
- euid – {type: UUID, optional: true}– идентификатор события
    1) meta – {type: object} – метаданные сообщения
- raw_answer – {type: array}– сообщение ассистента (response)

1) InfItem – {type: object} – элемент предназначен для отображения кликабельного текстового элемента, клик по которому приводит к автоматической отправке сообщения ассистенту

а) type – {type: string, optional: true}– userlink – фиксированное значение типа элемента

б) link – {type: string, optional: true}– текстовые данные для отображения

в) request – {type: string} – запрос, отправляемый ассистенту при клике по элементу. Если это поле отсутствует или не заполнено, то вместо него используется значение поля link

- HrefItem – {type: object}– элемент предназначен для отображения кликабельной ссылки

1) type – {type: string}– link – фиксированное значение типа элемента

2) ref – {type: string, optional: true}– текст для отображения. Если это поле отсутствует, то вместо него используется полный адрес ссылки

3) link – {type: string} – адрес документа (URL, Universal Resource Locator, универсальный указатель ресурсов), на который следует перейти при клике по элементу

4) target – {type: string, optional: true}– имя окна или фрейма, куда браузер будет загружать документ

- PreItem – {type: object}– элемент используется для указания, что текст внутри элемента должен отображаться с сохранением пробелов и переносов строк так, как они заданы изначально

1) type – {type: string} – pre - фиксированное значение типа элемента

2) text – {type: string}– преформатированный текст

- HtmlItem – {type: object}– элемент предназначен для начала новой строки в рамках ответа ассистента

1) type – {type: string}– br – фиксированное значение типа элемента

а) tag – {type: string, optional: true}– HTML-тег

- TextItem –{type: object} – элемент предназначен для отображения текстовых данных

1) type – {type: string} –text – фиксированное значение типа элемента

2) text – {type: string}– текстовые данные для отображения

3) parsed – {type: bool, optional: true} – обработанный текст. False - фиксированное значение

- ImgItem – {type: object}– элемент предназначен для отображения изображений в графическом формате gif, jpeg или png

1) type – {type: string}– img – фиксированное значение типа элемента
2) src – {type: string}– путь к графическому файлу
3) alt – {type: string}– альтернативный текст для изображения

- IntegrationItem – {type: object} – элемент предназначен для отображения информации об интеграции

1) type – {type: string}– dis - фиксированное значение типа элемента
2) name – {type: string} – имя интегрируемого сервиса
3) method – {type: string}– метод интегрируемого сервиса
4) arg – {type: object} – словарь, содержащий дополнительные аргументы, необходимые для выполнения интеграции

- ListItem – {type: array} – элемент предназначен для отображения набора маркированного списка

1) type – {type: string}– list – фиксированное значение типа элемента
2) rdered – {type: bool} –признак нумерованного списка. По умолчанию: true
3) items – {type: array} – массив элементов списка, где каждый элемент представляет собой JSON структуру со следующими полями:
   а) type – {type: string}– item – фиксированное значение типа элемента
   б) values –{type: array[response]}–массив элементов списка, представленных объектами типа response

- ButtonItem – {type: object}– элемент предназначен для отображения кнопки

1) type – {type: string}– button - фиксированное значение типа элемента
2) ref – {type: string, optional: true} – кнопка, отображаемая на экране
3) request – {type: string, optional: true}– запрос, отправляемый ассистенту при клике по элементу
4) link –{type: string, optional: true}–адрес документа (URL, UniversalResource Locator, универсальный указатель ресурсов), на который следует перейти при клике по элементу
5) target – {type: string, optional: true}– имя окна или фрейма, куда браузер будет загружать документ

5.7.1.2 Результаты

а) Успех

- HTTP Code: 200
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: true – фиксированное значение.
        б) result – {type: string}– результат обработки запроса.

б) Ошибка

- HTTP Code: 4** / 500
    1) Тело ответа представляет собой JSON структуру со следующими полями:
        а) success – {type: bool}: false – фиксированное значение.
        б) result – {type: object}:
        в) error_type – {type: string}– тип ошибки.
        г) error_message – {type: string} – описание ошибки.
        д) error_data – {type: string}– дополнительная информация об ошибке.

6 СЛУЖЕБНЫЕ МЕТОДЫ

6.1 Проверка состояния сервиса (/health_check)

Метод выполняет базовую проверку работоспособности приложения.

6.1.1.1 Запрос

a) Method

GET: {SAPI-DSN}/health_check

6.1.1.2 Результаты

a) Успех

HTTP Code: 200

6.2 Запуск/перезапуск сервиса (/startup)

Метод позволяет выполнить необходимые действия для инициализации сервиса.

6.2.1.1 Запрос

a) Method

GET: {SAPI-DSN}/startup

6.2.1.2 Результаты

a) Успех

HTTP Code: 200

6.3 Проверка готовности сервиса к обработке запросов(/readiness)

Метод предоставляет информацию о состоянии сервиса, включая готовность основных компонентов.

6.3.1.1 Запрос

a) Method

GET: {SAPI-DSN}/readiness

6.3.1.2 Результаты

a) Успех

HTTP Code: 200

6.4 Проверка активности сервиса и его компонентов (/liveness)

Метод используется для проверки жизнеспособности (лайвнесса) сервиса. Он помогает определить, активен ли сервис и работает ли он должным образом.

6.4.1.1 Запрос

a) Method

GET: {SAPI-DSN}/liveness

6.4.1.2 Результаты

a) Успех

HTTP Code: 200

6.5 Перенаправление к документации (/)

Метод представляет путь к документации.

6.5.1.1 Запрос

a) Method

GET: {SAPI-DSN}/

6.5.1.2 Результаты

a) Успех

HTTP Code: 200

6.6 Предоставление метрик (/metrics)

Метод используется для сбора и предоставления метрик и статистики работы сервиса.

6.6.1.1 Запрос

a) Method

GET: {SAPI-DSN}/metrics

6.6.1.2 Результаты

a) Успех

HTTP Code: 200

6.7 Поддерживаемые события (Events List)

Событие начала диалога (00b2fcbe-f27f-437b-a0d5-91072d840ed3)

Формат сообщений, отправляемых на установленный webhook (Response format)

Сообщения, передаваемые через Webhook, представляют собой JSON структуру одного из описанных ниже типов.

В каждом сообщении обязательно присутствуют следующие поля:

- type * - {type: string}: message – фиксированное значение.
- dialog_uid * – {type: uuid}– уникальный идентификатор диалога. *
- message * - {type: response}– сообщение для передачи собеседнику.
- context * - {type: context}– изменение контекста диалога.
- attachments * - {type: attarchments}- вложения
    1) files * - {type: array[file]}- структура

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Входящий номер сопроводительного документа и дата	Подпись	Дата
	изме-ненных	заме-ненных	новых	аннули-рованных					