

Утверждено 01.08.2024



Директор
ООО «Уралинновация»
И.А. Калинин

ООО «УРАЛИННОВАЦИЯ»

**Оmnikanальная платформа для автоматизированных
коммуникаций с клиентами TWIN**

Руководство пользователя (администратора компании)

2024 г.

Оглавление

1. Введение	5
2. Назначение и условия применения	5
3. Основные функции	6
4. Начало работы с личным кабинетом	6
4.1. Регистрация в личном кабинете	6
4.2. Вход в личный кабинет	7
4.3. Восстановление пароля	7
5. Обзор личного кабинета	8
5.1. Главная страница	8
5.2. Страница «Продвинутого поиска»	8
5.3. Раздел «Агенты NLU»	9
5.4. Раздел «Голосовые боты»	10
5.5. Раздел «Текстовые боты»	11
5.6. Раздел «Уведомления»	12
5.7. Раздел «Компании»	13
5.8. Раздел «Финансы»	14
5.9. Раздел «Отчеты»	14
6. Пошаговые инструкции	14
6.1. Управление агентами NLU	14
6.2. Управление намерениями	16
6.3. Управление сущностями	17
6.4. Редактор сценариев	18
6.5. Создание сценария бота	23
6.6. Тестирование сценария	23
6.7. Удаление сценария	24
6.8. Восстановление сценария из архива	24
6.9. Блоки редактора сценариев	24
6.9.1. Настройки	24
6.9.2. Информация	34
6.9.3. Вопрос	39
6.9.4. Результат	62
6.9.5. Пустой	68
6.9.6. Телепорт	70
6.9.7. Порядковый выбор	71
6.9.8. Случайный выбор	72
6.9.9. Запрос к серверу	73
6.9.10. Условие	79
6.9.11. Выражение	83
6.9.12. Пауза	84
6.10. Системные переменные в сценарии	88
6.10.1. Список системных переменных	88
6.11. Средства форматирования даты и времени	90

6.11.1. Список кодов форматирования	90
6.11.2. Склонение значений пользовательских переменных по падежам	93
6.12. Таймеры и принципы в распознавании речи	93
6.13. Активное слушание	94
6.14. Регулярные выражения	96
6.15. Управление заданиями на обзвон	100
6.15.1. Работа со списком заданий на обзвон	100
6.15.2. Поиск задания	101
6.15.3. Сортировка списка заданий	101
6.15.4. Фильтрация по статусу задания на обзвон	101
6.15.5. Редактирование задания на обзвон	102
6.15.6. Копирование идентификатора задания на обзвон	102
6.15.7. Запуск задания на обзвон	103
6.15.8. Звонки задания на обзвон	103
6.15.8. Удаление задания на обзвон	103
6.16. Создание задания на обзвон	103
6.17. Добавление кандидатов	107
6.18. Управление обзвоном	109
6.18.1. Запуск обзвона	109
6.18.2. Пауза	110
6.18.3. Отмена обзвона	110
6.18.4. Перезапуск вызовов	110
6.19. Работа со звонками в задании на обзвон	110
6.19.1. Просмотр списка звонков	110
6.19.2. Копирование ID звонка	111
6.19.3. Получение подробной информации о звонке	111
6.19.4. Прослушивание аудиозаписи звонка	111
6.19.5. Переход в редактор сценариев	112
6.20. Управление шаблонами	112
6.20.1. Создание шаблона задания на обзвон	112
6.20.2. Редактирование шаблона задания на обзвон	112
6.20.3. Удаление шаблона задания на обзвон	113
6.21. Поиск аудиозвонка по номеру телефона	Error! No bookmark name given.
6.22. Просмотр входящих и исходящих звонков	114
6.23. Добавление оператора связи	114
6.24. Добавление Caller ID внешней телефонии	116
6.25. Обработка входящих вызовов	117
6.26. Экспорт отчета по заданию на обзвон	118
6.27. Экспорт отчета по исходящим вызовам	119
6.28. Экспорт отчета по входящим вызовам	120
6.29. Экспорт отчета по телефонии с произвольными результатами	121
6.30. Выгрузка входящих звонков	122
6.31. Выгрузка исходящих звонков	122
6.32. Начало работы с BPL	123

6.33. Основные понятия BPL	124
6.33.1. Операции	124
6.33.2. Типы данных	130
6.33.3. Функции	133
6.33.4. Переменные	135
6.34. Функции BPL	137
6.34.1. Функции общего назначения	137
6.34.2. Математические функции	139
6.34.3. Строковые функции	142
6.34.4. Функции хэширования	144
6.34.5. Кодирование и декодирование	145
6.34.6. Работа с датой и временем	147
6.34.7. Управление очередью сообщений пользователя	150
6.34.8. Обработка сообщений бота	152
6.34.9. Взаимодействие с файлами	153
6.34.10. Операции с фактами	154
6.34.11. Функции таймера	157
6.34.12. Работа с текстом на естественном языке NLP	158
6.34.13. Понимание естественного языка	159
6.34.14. Функции для работы с HTTP	161
6.34.15. Системные функции	163
6.34.16. Функции GPT	163
6.34.17. Функции RekaAI	165
6.34.18. Функции GigaChat	166
6.34.19. Функции YandexGPT	167
6.34.20. Функции YCLIENTS	167
6.34.21. Объекты Request и Response	170
6.34.22. Объект FactQuery	173
6.34.23. Объект FactQueryCondition	174
6.34.24. Объект UserMessage	176
6.34.25. Объект Sentence	1776

1. Введение

Данное руководство предназначено для администраторов системы, которые хотят использовать все её возможности. В руководстве подробно описаны функции и возможности системы, а также приведены инструкции по использованию.

Платформа TWIN — это мощный инструмент, предназначенный для создания голосовых и чат-ботов. Она использует обучаемую языковую модель, что позволяет создавать интеллектуальные и адаптивные боты, способные понимать и обрабатывать естественный язык. Благодаря этому, роботы, созданные на платформе Twin, могут эффективно взаимодействовать с пользователями, предоставляя им точные и полезные ответы на их запросы.

Одной из ключевых особенностей робота, созданного на платформе TWIN, является его способность интегрироваться с другой платформой. Будь то Telegram, VK или другие мессенджеры, робот может быть легко адаптирован для работы в любой из этих средах. Это делает его универсальным инструментом, который может быть использован в самых разных контекстах и для самых разных целей.

2. Назначение и условия применения

Кабинет администратора предназначен для создания сценариев, а также управления заданиями на обзвон и клиентскими рассылками по SMS, электронной почте, в мессенджерах и социальных сетях.

Для использования кабинета администратора необходимо выполнить следующие условия:

- Иметь доступ к системе.
- Знать логин и пароль для входа в личный кабинет.
- Быть зарегистрированным администратором системы.

В кабинете администратора можно выполнять следующие функции:

- Создание сценариев для голосовых вызовов и клиентских рассылок.
- Настройка телефонии для выполнения голосовых вызовов.
- Регистрация зависимых компаний.
- Создание шаблонов и управление рассылками в мессенджерах и социальных сетях.
- Просмотр статистики по использованию сервиса.
- Создание отчётов.

С помощью этих функций администратор может эффективно управлять системой и обеспечивать её бесперебойную работу.

3. Основные функции

Основные функции кабинета администратора включают:

1. Создание сценариев для голосовых вызовов и клиентских рассылок:
 - Выбор типа сценария (голосовой вызов, SMS-рассылка, email-рассылка и т. д.).
 - Ввод текста сценария.
 - Сохранение сценария.
2. Настройка телефонии для выполнения голосовых вызовов:
 - Определение номера телефона, с которого будут выполняться вызовы.
 - Настройка параметров вызова (время ожидания ответа, количество попыток дозвона и т. п.).
3. Регистрация зависимых компаний:
 - Добавление новых компаний в систему.
 - Управление данными о компаниях (название, адрес, телефон и т. д.).
4. Создание шаблонов и управление рассылками в мессенджерах и социальных сетях:
 - Создание шаблона рассылки.
 - Привязка шаблона к сценарию.
 - Отправка рассылки по выбранному каналу связи.
5. Просмотр статистики по использованию сервиса:
 - Анализ данных о выполненных звонках и отправленных рассылках.
 - Оценка эффективности работы системы.
6. Создание отчётов:
 - Формирование отчётов по различным параметрам (количество звонков, длительность разговоров и т. п.).
 - Экспорт отчётов в формате CSV или PDF.

4. Начало работы с личным кабинетом

4.1. Регистрация в личном кабинете

1. Перейдите на [главную страницу](#) платформы и нажмите кнопку **Регистрация** в правом верхнем углу.
2. В появившемся окне заполните поля:
 - **Имя и фамилия:** введите ваше имя и фамилию.
 - **Электронная почта:** укажите действующий почтовый адрес.

- **Контактный телефон:** укажите ваш действующий номер телефона вместе с кодом страны. Например, +79001112200.
- **Пароль:** придумайте пароль.

Минимальная длина пароля — 8 символов. Пароль должен содержать:

- прописные и строчные буквы латинского алфавита;

- цифры и специальные символы: ? ! ~ @ # \$ % ^ & _ - + * = ; : , . / \ | ` [] { } () .

3. Дайте согласие на условия пользовательского соглашения и политику обработки персональных данных.

4. Нажмите кнопку **Далее**. Откроется окно подтверждения учетной записи. На указанный при регистрации номер телефона будет отправлен защитный код.

Если СМС не приходит или срок действия кода из СМС истек, убедитесь, что номер телефона указан верно и запросите код повторно, нажав кнопку **Отправить сообщение еще раз**, которая появится через 3 минуты после отправки кода.

5. Введите код и нажмите **Готово**.

4.2. Вход в личный кабинет

Для авторизации в системе введите адрес электронной почты и пароль. Для входа в личный кабинет нажмите кнопку **Войти в кабинет**.

Чтобы сохранить введенные данные, установите флажок **Запомнить меня**.

Если вы забыли пароль от личного кабинета, пройдите процедуру восстановления пароля. Для этого:

1. Нажмите **Забыли пароль?**
2. Введите адрес электронной почты, на которую был зарегистрирован личный кабинет.
3. Нажмите **Отправить**. На указанный адрес электронной почты будет направлено письмо с инструкцией для восстановления пароля.

4.3. Восстановление пароля

Если вы забыли пароль от личного кабинета, вы можете в любое время его восстановить. Для этого выполните следующие действия:

1. Перейдите на страницу авторизации.
2. Пройдите по ссылке **Забыли пароль**.

3. На странице восстановления пароля укажите адрес вашей электронной почты и нажмите кнопку **Восстановить**. На указанную почту будет отправлено письмо с ссылкой на сброс пароля.
4. Пройдите по ссылке и на открывшейся странице задайте новый пароль.

5. Обзор личного кабинета

5.1. Главная страница

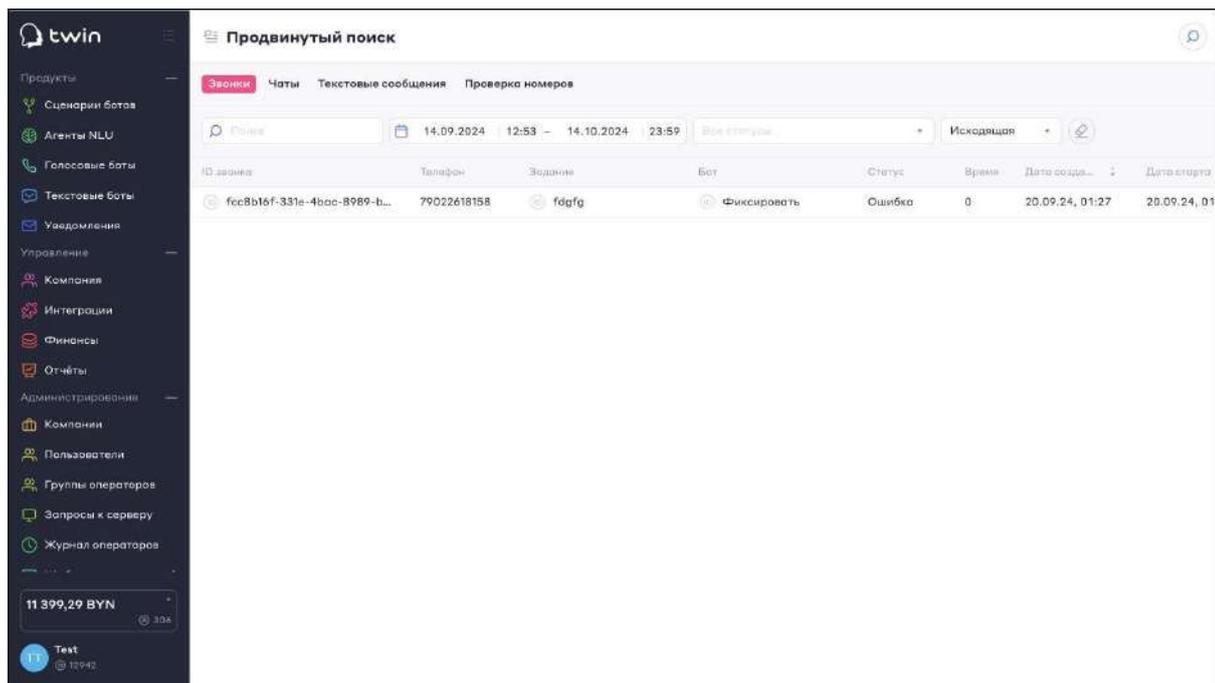
Главное окно личного кабинета администратора состоит из строки состояния, бокового меню и рабочей области.

В правом нижнем углу отображается информация о текущем состоянии счета пользователя, имя и идентификационный номер администратора, название и идентификационный номер организации. При нажатии на область с информацией о состоянии счета, а также область с идентификаторами администратора и организации отображается меню со списком команд, при помощи которых можно быстро сформировать счет на оплату или перейти на страницу настроек личного кабинета пользователя.

5.2. Страница «Продвинутого поиска»

Кнопка для перехода на страницу продвинутого поиска находится в правом верхнем углу. При нажатии на элемент управления со значком лупы в правом верхнем углу окна отображается страница продвинутого поиска, на которой можно искать диалоги в чатах, текстовые сообщения в различных каналах, проверять номера и звонки.

На странице «Продвинутый поиск» вы можете найти нужную информацию по использованию сервисов компании TWIN (например, информацию по чатам, текстовым сообщениям и др.).



Чтобы перейти на страницу поиска, выполните щелчок по кнопке с пиктограммой лупы в правом верхнем углу личного кабинета.

Откроется страница «Продвинутый поиск». В верхней части страницы расположены поля, при помощи которых вы можете выполнять поиск и фильтрацию отображаемой информации по сервисам платформы.

По умолчанию на странице отображается информация по сервису текстовых сообщений (клиентским рассылкам).

На странице можно выбрать тип сервиса, для которого будет отображаться информация на странице. Значение в этом поле выбирается при помощи выпадающего списка. Для выбора доступны следующие варианты:

- Чаты;
- Текстовые сообщения;
- Проверка номеров;
- Звонки.

5.3. Раздел «Агенты NLU»

NLU — система для понимания естественного языка в чат- и голосовых ботах. Играет ключевую роль в диалоговых платформах и искусственном интеллекте.

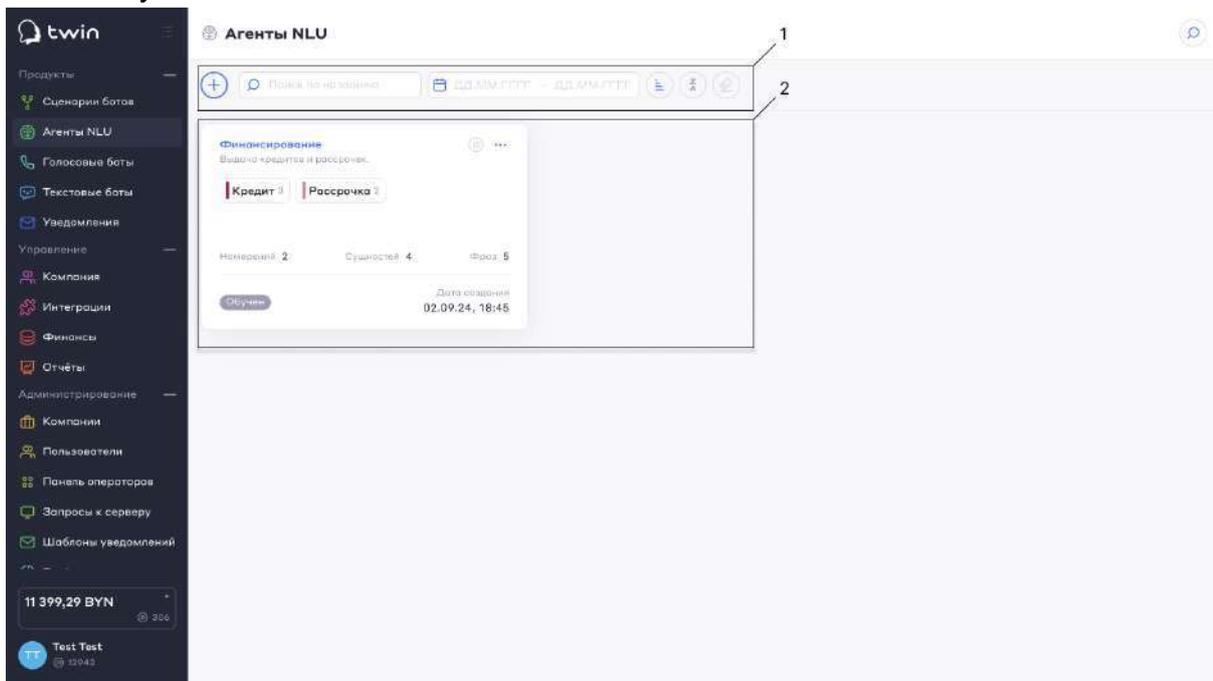
В разделе «Агенты NLU» личного кабинета можно:

1. Создавать и настраивать агентов NLU, добавлять намерения и фразы для их обучения.
2. Управлять сущностями (определенными значениями, такими как даты, имена и т. д.), которые помогают улучшать понимание.

3. Тестировать агентов для проверки работы и точности распознавания фраз.
4. Интегрировать NLU-агентов в сценарии ботов, делая их более интерактивными.

В разделе «Агенты NLU» на изображении:

1. **Фильтры и сортировка.** Панель позволяет отфильтровать агентов по дате создания и использовать сортировку для упрощенного поиска.
2. **Список агентов.** Включает карточки агентов NLU с основной информацией: название, ключевые сущности и намерения, дата создания и статус обучения.



5.4. Раздел «Голосовые боты»

В разделе «Голосовые боты» личного кабинета можно:

1. Создавать и настраивать сценарии для ботов, в том числе управлять заданиями на обзвон.
2. Управлять списками клиентов для обзвона и выгружать отчеты по результатам.
3. Настраивать телефонию для интеграции звонков и управлять шаблонами диалогов.
4. Диагностировать ошибки и получать советы по устранению неполадок.

Раздел состоит из следующих элементов:

1. **Вкладки раздела:** включает вкладки **Задания**, **Шаблоны заданий**, **Настройки** и **Детализация**. Каждая из них позволяет переходить к

различным частям работы с ботами, таким как управление задачами обзвона, настройка шаблонов и просмотр детальной статистики.

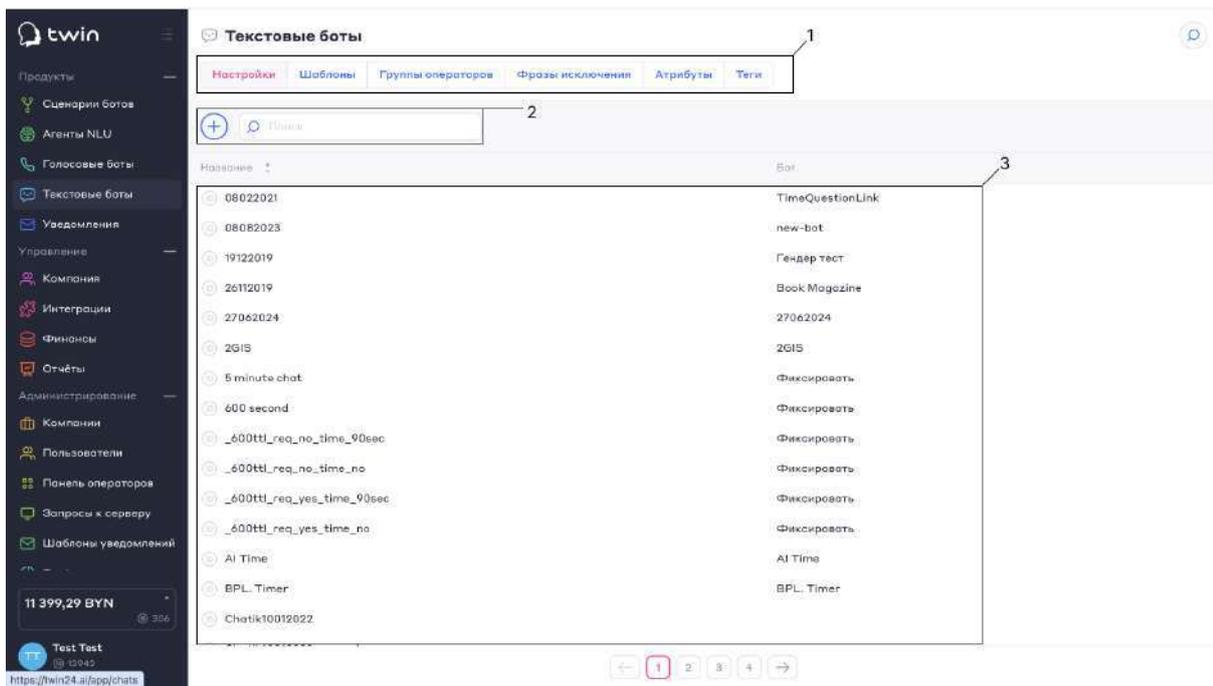
2. **Фильтры и поиск:** позволяет искать задания, выбирать определенные параметры и настраивать временные рамки. Это облегчает фильтрацию заданий по критериям.
3. **Список голосовых ботов:** содержит список всех заданий с подробной информацией по каждому — название, прогресс, статус, режим запуска и даты создания/обновления.

Название	CPS	Кандидаты	Номера	Прогресс	Статус	Режим запуска	Создано	Обновлено
fdgfg	1	1	1	100%	Выполнено	Вручную	20.09.24, 01:26	20.09.24, 01:26
test-doc	1	0	0	0%	Создано	Вручную	02.09.24, 19:15	02.09.24, 19:15
06/06/24 test 2	1	2	2	100%	Остановлено	Вручную	06.06.24, 14:42	06.06.24, 14:42
06/06/24 test	1	0	0	0%	Остановлено	Вручную	06.06.24, 14:39	06.06.24, 14:39
12092023_001	1	2	2	100%	Выполнено	Вручную	13.09.23, 03:32	12.11.23, 16:53
test0708	1	0	0	0%	Остановлено	Вручную	08.08.23, 04:48	01.11.23, 16:53
2703	1	1	1	100%	Остановлено	Вручную	27.03.23, 20:33	01.11.23, 16:53
8119002	1	3	3	100%	Выполнено	Вручную	17.11.23, 19:10	17.11.23, 18:48
8119001	1	4	4	100%	Выполнено	Вручную	17.11.23, 18:48	17.11.23, 18:17
TWIN-8119	1	4	4	75%	Пауза	Вручную	17.11.23, 18:17	17.11.23, 16:09
1711001	1	2	2	100%	Выполнено	Вручную	17.11.23, 16:09	20.03.23, 16:18
Какой то новый коллtask 2	1	0	0	0%	Создано	Вручную	20.03.23, 16:18	20.03.23, 16:16
Какой то новый коллtask	1	0	0	0%	Создано	Вручную	20.03.23, 16:16	17.03.23, 22:06
Тестовый коллtask	1	0	0	0%	Создано	Вручную	17.03.23, 22:06	17.03.23, 21:55
callTaskv1703001	1	0	0	0%	Создано	Вручную	17.03.23, 21:55	17.03.23, 21:49
Test1703_001	1	0	0	0%	Создано	Вручную	17.03.23, 21:49	16.01.23, 16:53
новый колл task	1	0	0	0%	Создано	Вручную	16.01.23, 16:53	

5.5. Раздел «Текстовые боты»

В разделе личного кабинета текстовых ботов на платформе TWIN пользователи могут управлять чатами, создавать и настраивать текстовых ботов для поддержки и общения с клиентами. В интерфейсе личного кабинета можно:

1. **Создавать и редактировать ботов** – позволяет создавать нового бота или изменять существующего, задавая логику взаимодействия.
2. **Настраивать виджет** – параметры внешнего вида и поведения виджета чата на сайте.
3. **Просматривать отчеты** – раздел аналитики с данными об эффективности взаимодействий с пользователями.



5.6. Раздел «Уведомления»

В разделе можно выполнять рассылку сообщений клиентам, используя различные каналы связи.

Основные функции сервиса уведомлений:

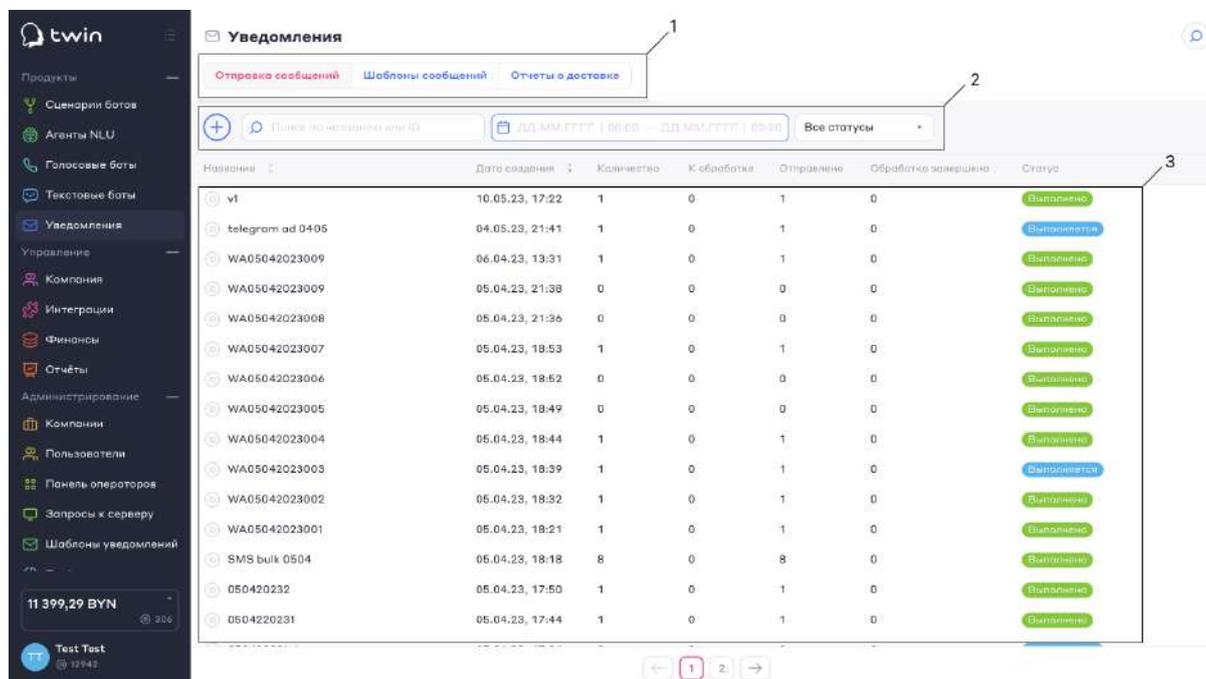
1. Отправка текстовых уведомлений пользователям.
2. Отложенная отправка (сообщение отправится в установленное время). При необходимости можно отменить отправку сообщения до наступления установленного времени. Благодаря этому можно контролировать, что сообщение не придет пользователю в неположенное время (например ночью).
3. Сокращение ссылок. Длинные ссылки сокращаются для уменьшения стоимости отправки сообщения. Это касается только канала SMS.
4. Использование шаблонов. Чтобы не передавать каждый раз весь массив данных уведомления через API, можно создать шаблон, описывающий всю структуру сообщения, и менять только номер телефона или электронный адрес получателя и переменные.
5. Отправка уведомлений по различным каналам: SMS, e-mail, push, WhatsApp, Viber, Telegram, V Kontakte, Odnoklassniki, chat. По умолчанию сообщения отправляются последовательно: если сообщение было доставлено по одному из каналов, то по остальным каналам оно не отправляется.

В разделе «Уведомления» личного кабинета можно:

1. Создавать задания для отправки уведомлений клиентам.

2. Управлять шаблонами уведомлений — создавать, редактировать и удалять их.
3. Просматривать отчеты об отправленных уведомлениях, анализировать статус доставки и эффективность.

Интерфейс состоит из вкладок для настройки шаблонов, создания новых задач и управления историей уведомлений.



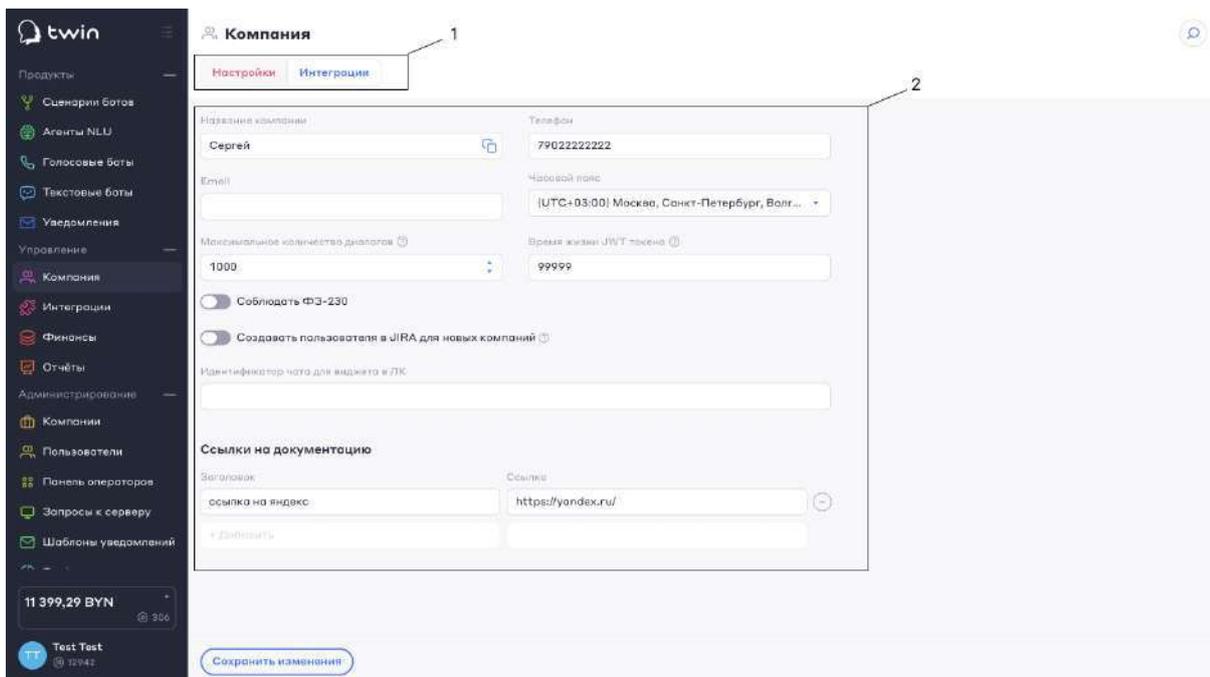
5.7. Раздел «Компании»

В разделе Компании отображается финансовая информация об организации реселлера, а также всех его зависимых компаний.

Для перехода на данную страницу выберите в боковом меню пункт **Компании**.

Раздел состоиз из:

1. Вкладки «Настройки» и «Интеграция» — они находятся в верхней части раздела и позволяют переключаться между настройками компании и интеграциями с другими системами.
2. Область настроек компании — основная часть окна, где отображаются поля для ввода информации о компании. Здесь можно указать название, телефон, часовой пояс, максимальное количество диалогов, срок действия токена, а также включить создание пользователей в JIRA и соблюдение ФЗ-230. Раздел также поддерживает добавление ссылок на документацию.



5.8. Раздел «Финансы»

Раздел «Финансы» в системе предназначен для управления балансом компании, оплаты услуг и контроля расходов. Здесь можно пополнить баланс, просматривать расходы, управлять счетами, включая их фильтрацию, сортировку, скачивание в PDF, а также получать ежемесячные отчеты по платежам. Раздел также позволяет рассчитать стоимость использования токенов для моделей ChatGPT.

5.9. Раздел «Отчеты»

В этом разделе можно получить детализированную информация о стоимости сервисов и услуг платформы для всех зависимых компаний реселлера. Зависимыми называются компании, с которыми реселлер компании TwiN заключил договор о сотрудничестве.

6. Пошаговые инструкции

6.1. Управление агентами NLU

NLU — это инструмент, который мы обучаем и используем для анализа фраз от людей, их реакций на те или иные события, их поведения, целей и намерений.

Подготовительные шаги

1. Зарегистрируйтесь и пополните баланс компании.
2. Создайте бота, в который будет интегрирован NLU.

Создание агента NLU

1. Нажмите **Создать** агента или  в левом верхнем углу. Откроется окно настроек.
2. **Название:** придумайте название нового агента.

□ Для названия используйте только латинские буквы без пробелов и начинайте название с заглавной буквы.

3. **Описание** — короткое описание на русском языке.
4. В блоке **Обработка** текста включите нужный переключатель, которые будут отвечать за основную систему распознавания намерений во фразах клиентов:
 - **Лемма** — чтобы приводить каждое слово в каждом обучающем примере к его начальной форме (например, для существительных начальной является форма единственного числа именительного падежа). Рекомендуется к установке – тогда при наполнении NLU-агента вам не придется указывать фразы, содержащие одни и те же слова со всеми возможными окончаниями.
 - **Стемма** — чтобы удалять в каждом слове в каждом обучающем примере окончания и некоторые простые суффиксы таким образом, что от каждого слова остается только его основа.
 - Исправление опечаток — чтобы исправлять в словах опечатки с учетом контекста и состава слова. Не рекомендуется выставлять, если в агенте используются специфические термины.
5. В блоке **Дополнительно:**
 - **Конфигурация обучения:** выберите из выпадающего списка конфигурацию.
 - **Активность:** по умолчанию переключатель активен. Чтобы в дальнейшем интегрировать агента в сценарии оставьте переключатель активным.
 - **Порог точности распознавания:** задайте процент совпадения фразы клиента с фразой из наполнения намерения. Чтобы установить этот порог, включите переключатель и переместите ползунок от 0 до 100. Число, которое вы выберете, будет отображаться слева от шкалы. При помощи данного параметра можно управлять гибкостью и точностью доверия к предположениям системы о том, к чему относится сказанная или написанная клиентом фраза. Порог точности может иметь различные значения в зависимости от важности ответа клиента на тот или иной вопрос. Выбранный порог применяется во всех сценариях, в которых используется NLU-агент.

□ Чем больше порог для точности, тем осторожнее робот.

6. Нажмите кнопку **Сохранить**. Созданный вами агент появится на странице раздела, где будет указан статус и дата создания.

Редактирование агента NLU

Чтобы изменить параметры агента, нажмите на  в блоке агента и внесите нужные изменения. Нажмите **Сохранить**.

Удаление агента NLU

Удалите ненужного агента:

1. Нажмите на  в в блоке нужного агента.

2. Нажмите **Удалить**. На экране отобразится сообщение с подтверждением удаления.
3. Нажмите кнопку **Удалить**.

6. 2. Управление намерениями

Намерения — это то, что пользователь хочет получить при общении с ботом.

Цель/смысл его обращения к боту, какую проблему он хочет решить или что именно он хочет узнать.

Подготовительные шаги

1. **Зарегистрируйтесь и пополните баланс компании.**
2. **Создайте бота, в который будет интегрирован NLU.**
3. **Создайте NLU-агента, чтобы добавить новое намерение в агент.**

Создание намерений

1. Нажмите на ранее созданного агента NLU. Откроется страница создания намерений.
2. Нажмите **Создать** намерение или  в левом верхнем углу.
3. Задайте параметры намерения в открывшемся окне:
 - **Название** — краткое название намерения.

Название должно быть уникальным и содержать только латинские буквы, нижнее подчеркивание и цифры.

- **Цвет** — цвет метки намерения. С помощью меток можно быстро отличать намерения друг от друга.
- **Описание** — полное имя намерения.
- Нажмите кнопку **Сохранить**. В словаре намерений появится новая запись.

Чем больше намерений вы добавите, тем точнее робот сможет определять желания клиентов.

Редактирование намерения

Чтобы изменить параметры намерения, нажмите на  в строке намерения и внесите нужные изменения. Нажмите **Сохранить**.

Наполнение намерений обучающими фразами

После создания намерения наполните его обучающими фразами, которые в дальнейшем будут ассоциироваться с ним. Для этого нажмите в строке намерения.

Откроется окно добавления фраз в намерении.

Добавление обучающих фраз вручную:

1. Введите в поле для поиска фразу.
2. Нажмите **Добавить** или **Enter** на клавиатуре компьютера.

Импорт обучающих фраз в намерение из файла:

1. Нажмите **Импорт фраз**.
2. Выберите файлы для импорта с вашего компьютера. Агент может принимать файлы формата *.txt (одна фраза на одной строке) и *.csv.
3. Нажмите **Открыть**. Все фразы из файла автоматически появятся в намерении.

Обучение агента

После наполнения NLU-агента фразами обучите его. Рекомендуем делать это каждый раз после внесения любых изменений, так как чем больше его содержание, тем дольше проходит обучение. Только после этого вы сможете использовать его в сценариях.

Чтобы начать обучение нажмите **Обучить** в правом верхнем углу.

Удаление намерения

Удалите ненужное намерение:

1. Нажмите на меню в строке нужного намерения.
2. Нажмите **Удалить**.
3. Нажмите кнопку **Удалить**, чтобы подтвердить удаление.

6.3. Управление сущностями

Сущности — это инструмент, который позволяет вычленять ключевые слова в ответе клиента: конкретные факты и определенные уточнения, которые имеют одинаковый контекст в рамках одного намерения.

Подготовительные шаги

1. **Зарегистрируйтесь** и пополните баланс компании.
2. Создайте бота, в который будет интегрирован NLU.
3. Создайте NLU-агента, чтобы добавить новое намерение в агент.
4. Создайте намерения.

Создание сущности

1. Нажмите на ранее созданного агента NLU. Откроется страница создания намерений.
2. Перейдите на вкладку **Сущности**.
3. Нажмите **Создать сущность** или .
4. Задайте параметры намерения в открывшемся окне:
 - **Имя сущности** — уникальное название сущности. Введите имя сущности на латинице.
 - **Цвет** — цвет маркера сущности. С помощью цвета маркера можно отличать сущности друг от друга в списке.
 - **Описание** — короткое описание сущности на русском языке.
5. Нажмите **Сохранить**.

Редактирование сущности

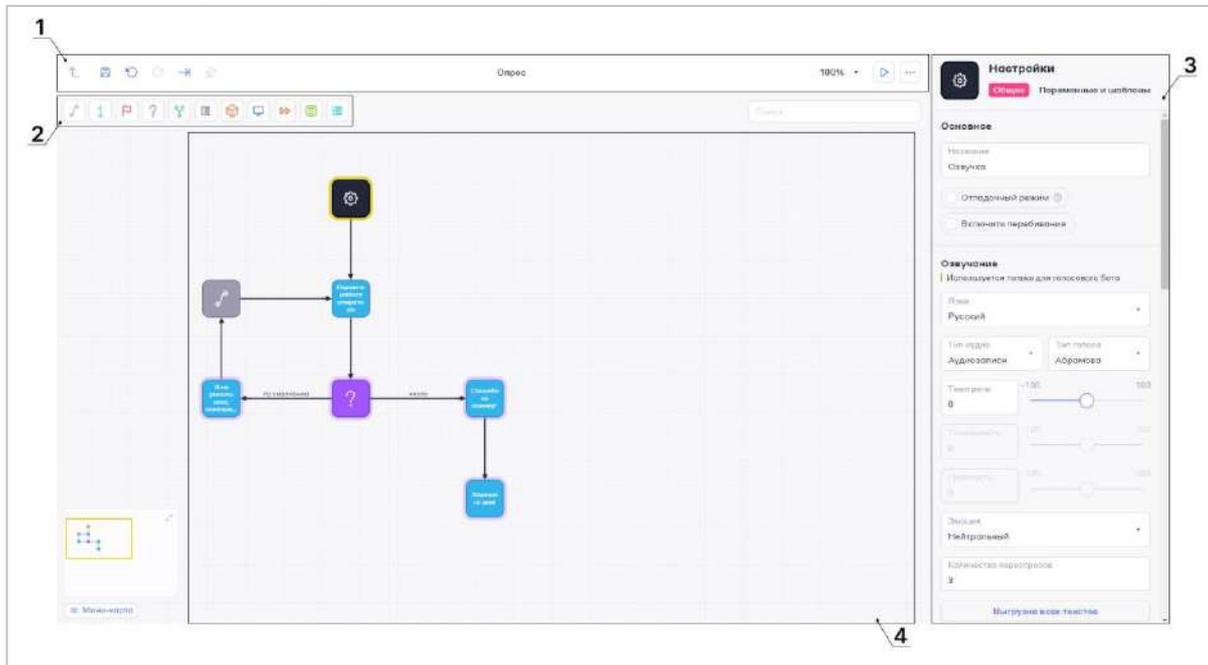
Чтобы изменить параметры сущности, нажмите на иконку редактирования в строке сущности и внесите нужные изменения. Нажмите **Сохранить**.

Удаление сущности

1. Нажмите на меню в строке сущности.
2. Нажмите **Удалить**. На экране отобразится сообщение с подтверждением удаления.
3. Нажмите кнопку **Удалить**.

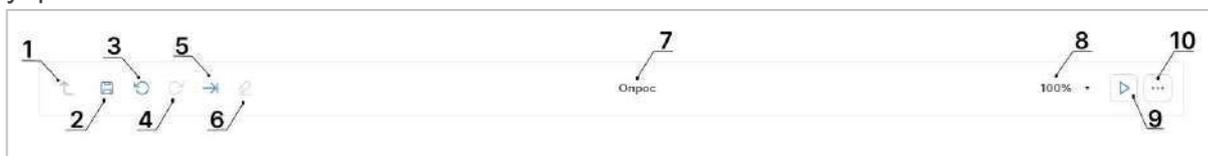
6.4. Редактор сценариев

Компоненты редактора сценариев



1. Панель управления

Для эффективной навигации и тестирования сценариев воспользуйтесь панелью управления.



1. Возврат в список сценариев.

❑ **Сохраните сценарий перед выходом, чтобы не утратить изменения.**

2. Сохранение изменений. Если в свойствах блока есть ошибки, блок с ошибкой будет подсвечен, а сценарий не сохранится.
3. Отмена выполненного действия.
4. Возврат отмененного действия.
5. Переход к последнему добавленному блоку.
6. Стереть выбранные (выделенные) в рабочей области элементы.

❑ **Стартовый блок Настройки удалить невозможно. Подробнее о блоке читайте в статье [Настройки](#).**

7. Название сценария.
8. Изменение масштаба сценария в рабочей области.
9. Выбор режима и переход к тестированию сценария:
 - Текстовый режим;
 - Голосовой ввод.
10. Прочие настройки:
 - Создать копию сценария — создать дубликат сценария и всех агентов, которые в нем участвуют.
 - Управление медиа — поиск, прослушивание, скачивание, удаление аудиофайлов, которые используются в сценарии.
 - Настройки сценария — переход к настройкам стартового блока.
 - Помощь — инструкция по работе с редактором сценариев.

2. Панель инструментов

Панель содержит блоки для визуального создания сценариев в рабочей области редактора.

3. Панель свойств

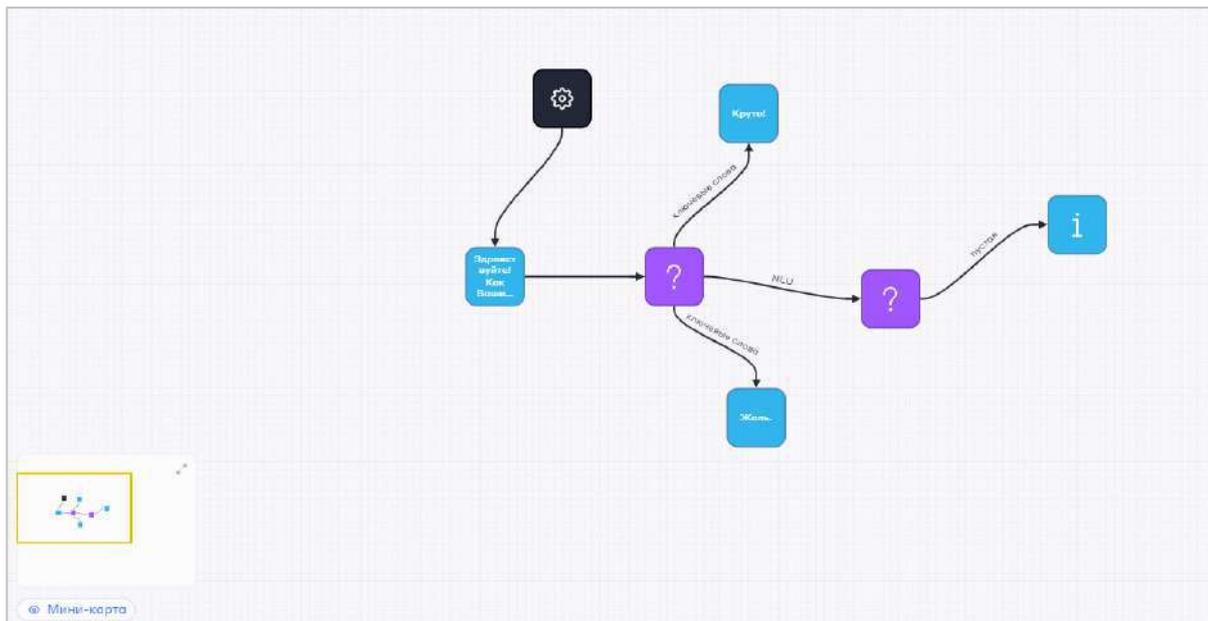
Панель содержит элементы управления, при помощи которых задаются свойства блоков и стрелок.

4. Рабочая область

Рабочая область — это пространство, в котором создаются сценарии из блоков и стрелок.

Для удобства работы с пространством доступны различные способы перемещения по области:

- **Колесо прокрутки** — перемещает рабочую область вертикально. Если зажать клавишу Shift и крутить колесо, то перемещение происходит горизонтально.
- **Стрелки на клавиатуре** — клавиши вверх, вниз, влево и вправо позволяют перемещаться по рабочей области.
- **WASD** — альтернативные клавиши для перемещения аналогично стрелкам.
- **Средняя кнопка мыши** — зажав и удерживая среднюю кнопку мыши, перемещайте рабочую область в любом направлении.
- **Ctrl + левая кнопка мыши** — аналогично перемещению средней кнопкой, зажмите Ctrl и левую кнопку мыши для перемещения.
- **Миникарта** — в нижнем левом углу рабочей области расположено мини-представление области, на котором отображается текущий видимый участок. Чтобы переместиться по рабочей области с помощью миникарты, зажмите видимую область левой кнопкой мыши и двигайте ее.



Элементы сценария

Двумя основными элементами, из которых строится сценарий, являются блоки и стрелки. Сценарий состоит из блоков, которые связываются при помощи стрелок. При первом заходе в сценарий отображается только стартовый блок.

Блоки

Блоки — это главные компоненты сценария, «кирпичики», из которых строится сценарий.

Название блока	Изображение блока	Название блока	Изображение блока
Настройки		Случайный выбор	
Информация		Запрос к серверу	
Вопрос		Условие	
Результат		Выражение	
Пустой		Пауза	

Телепорт			
Порядковый выбор			

Подробно про каждый блок читайте в разделе [Блоки редактора сценариев](#).

Добавление блока в сценарий

Из панели инструментов:

1. Нажмите на значок блока на панели инструментов.
2. Переместите блок на рабочую область, удерживая левую кнопку мыши. К добавленному блоку необходимо провести стрелку вручную (см. ниже раздел Стрелки).

Из блока:

1. Нажмите на блок, из которого будет вытягиваться следующий блок.
2. Зажмите значок нужного блока и тяните его в рабочую область.
3. Отпустите блок, когда он установится на нужное место.

Данный блок уже соединен с предыдущим блоком стрелкой и добавлять к нему стрелку не нужно.

Из стрелки:

Блок может быть вставлен из стрелки, которая тянется к другому блоку – тогда новый блок встанет между стрелкой и этим блоком.

Для этого выполните следующие шаги:

1. Нажмите на стрелку, из которой должен тянуться новый блок.
2. Нажмите на иконку кубика рядом с заголовком в открывшемся окне свойств стрелки. Выберите блок из выпадающего списка.

Смена типа блока

Не распространяется на стартовый блок.

Чтобы преобразовать блок любого типа в блок другого типа:

1. Нажмите на нужный блок, появится панель свойств блока.
2. Нажмите на  возле названия типа блока.
3. Выберите нужный тип блока.

При смене типа блока все данные, связанные с этим типом блока, будут утеряны.

Копирование блока

Windows, Linux:

1. Нажмите на блок или выделите несколько блоков, которые хотите скопировать, в рабочей области.
2. Нажмите **Ctrl + C** на клавиатуре.
3. Вставьте блок, нажав **Ctrl + V** на клавиатуре. Скопированные блоки появятся на рабочей области.

macOS:

1. Выделите блоки, которые нужно скопировать.
2. Нажмите **Command + C** на клавиатуре.
3. Нажмите **Command + V** на клавиатуре. Скопированные блоки появятся на рабочей области.
- 4.
5. Нажмите на блок или выделите несколько блоков, которые хотите скопировать, в рабочей области.
6. Нажмите **Ctrl + C** на клавиатуре.
7. Вставьте блок, нажав **Ctrl + V** на клавиатуре. Скопированные блоки появятся на рабочей области.

Удаление блока

1. Нажмите на блок или выделите несколько блоков в рабочей области.
2. Нажмите клавишу **Delete (Del)** на клавиатуре. Блоки будут удалены вместе со входящими в них и исходящими из них стрелками.

Стрелки

Стрелки — условия, от выполнения которых зависит то, как развивается диалог бота с клиентом.

Стрелки соединяют блоки сценария, показывают направление сценария и определяют условия перехода от одного блока к другому.

□ Если блок в сценарии не связан стрелками с другими блоками, такой блок не будет участвовать в диалогах.

Каждая стрелка обладает уникальными свойствами.

Соединение блоков стрелкой

1. Установите два блока: из которого будет вытягиваться стрелка (блок 1) и в который будет тянуться стрелка (блок 2).
2. Вытяните стрелку, используя один из двух вариантов:
 - Нажмите на блок 1, затем на значок стрелки (пустого блока) и протяните ее в блок 2.
 - Нажмите на блок 1, зажмите клавишу **Shift** на клавиатуре и нажмите на блок 2.

Определение начала и конца стрелки

Чтобы узнать из какого блока выходит и в какой блок ведет стрелка в сценарии бота:

1. Нажмите на интересующую стрелку.
2. Нажмите  в свойствах стрелки, чтобы узнать из какого блока выходит стрелка.
3. Нажмите  в свойствах стрелки, чтобы узнать в какой блок ведет стрелка.

Удаление стрелки

Нажмите на стрелку в рабочей области и нажмите клавишу **Delete (Del)** на клавиатуре.

6.5. Создание сценария бота

1. Перейдите в раздел **Сценарии ботов** [личного кабинета](#).
2. Нажмите кнопку **Создать сценарий** или  в левом верхнем углу.
3. Введите название сценария. В качестве имени сценария используйте простые и понятные слова и фразы, которые точно раскрывают его назначение. В дальнейшем название поможет вам найти сценарий в списке.
4. Создайте сценарий, используя [инструменты редактора сценариев](#).
5. Нажмите **Сохранить** для добавления сценария в список.

6.6. Тестирование сценария

1. Перейдите в раздел Сценарии ботов [личного кабинета](#).
2. Откройте нужный сценарий.

Текстовый бот:

3. Нажмите  в правой части верхней панели.
4. Нажмите Текстовый режим. Откроется окно тестирования бота.
5. Отправьте боту сообщение, чтобы начать диалог в текстовом режиме.
6. Тщательно и подробно протестируйте сценарий. Убедитесь, что любые варианты хода диалога, в том числе неочевидные, обрабатываются корректно.

Синтез речи:

3. Нажмите  в правой части верхней панели.
4. Нажмите Голосовой ввод. Откроется окно тестирования бота.
5. Наведите курсор на сообщение, затем нажмите , чтобы воспроизвести сообщение.
6. Тщательно и подробно протестируйте сценарий. Убедитесь, что любые варианты хода диалога, в том числе неочевидные, обрабатываются корректно.

При тестировании сценария в голосовом режиме используется [упрощенная версия ASR](#).

Виджет:

Чтобы протестировать работу сценария в виджете, создайте тестовый виджет.

6.7. Удаление сценария

При удалении сценария бота, он не удаляется безвозвратно, а перемещается в архив. Это позволяет восстановить ранее удаленный сценарий бота в будущем.

Чтобы перенести сценарий в архив, выполните следующие шаги:

1. Перейдите в раздел **Сценарии ботов**.
2. Нажмите на  в строке сценария, который вы хотите удалить.
3. Нажмите кнопку **Удалить**.
4. Подтвердите удаление. После сценарий будет перемещен в архив.

6.8. Восстановление сценария из архива

1. Перейдите в раздел **Сценарии ботов**.
2. Нажмите **Архив** в правом верхнем углу. Откроется окно со списком сценариев, помещенных в архив.
3. Отсортируйте результаты поиска: по названию, по дате удаления. По умолчанию сценарии располагаются в порядке убывания даты удаления. Это значит, что последний удаленный сценарий находится вверху списка.
4. Нажмите в строке нужного сценария. После этого сценарий вернется в раздел **Сценарии ботов**, где вы сможете продолжить работу над ним.
5. Нажмите **Заккрыть**, чтобы выйти из архива.

6.9. Блоки редактора сценариев

6.9.1. Настройки

Блок **Настройки** обозначает начало сценария, с него начинается присоединение блоков. В сценарии может быть только один стартовый блок, и его нельзя удалить. Он служит как ориентир, облегчая поиск начала в сложных сценариях с большим числом блоков. В стартовом блоке настраиваются параметры, которые будут применяться ко всему сценарию.

Из блока можно провести стрелку только в один блок.

Свойства блока

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Свойство	Описание
Общее	
Основное	
Название	<p>Название сценария.</p> <p><input type="checkbox"/> Название сценария может быть отредактировано только в данном блоке, оно не редактируется в списке сценариев.</p>
Отладочный режим	<p>Включается по просьбе технической поддержки и используется для выявления и устранения ошибок в сценарии. После установки флажка будет запущено подробное логирование действий робота для передачи разработчикам отладочных данных.</p> <p>После включения отладочного режима повторите диалог, в котором возникают проблемы. Все ошибки, возникающие при отработке сценария, будут зафиксированы на сервере разработчика.</p> <p><input type="checkbox"/> Использование отладочного режима увеличивает вычислительную нагрузку на систему, поэтому не рекомендуется включать данный режим без необходимости.</p>
Включить перебивания	<p>Установите флажок, чтобы включить режим прерывания бота в диалогах. Подробнее о перебиваниях читайте в статье Перебивания.</p> <p><input type="checkbox"/> Работает только в телефонии.</p>

Озвучание

Используется только для голосовых ботов.

Язык	Язык речи, который распознается системой. Список доступных языков.	
Тип аудио	Вид голосовых сообщений робота. Доступны следующие:	
	Синтез	Синтез речи по указанной фразе.
	Аудиозаписи	Воспроизведение предварительно подготовленных аудиофайлов, прикрепленных к блокам сценария. Если к некоторым блокам по каким-то причинам не прикреплены аудиофайлы с озвучкой, то текст будет озвучен синтезом.
Тип голоса	Тип голоса робота в диалогах с клиентом при синтезе речи. В зависимости от выбранного голоса набор настроек может меняться.	
Темп речи	Скорость речи. Может варьироваться от медленного и разборчивого до быстрого и бодрого, в зависимости от контекста и настроения диалога.	
Тональность	Тон голоса. Чем ниже значение, тем ниже звучит голос; чем выше значение, тем выше голос.	
Громкость	Громкость речи. Может варьироваться от тихого шепота до громкого и отчетливого произношения слов.	

Эмоция	<p>Эмоциональный оттенок речи. Возможные варианты:</p> <ul style="list-style-type: none"> ● нейтральный, ● доброжелательный, ● раздраженный.
Количество переспросов	<p>Сколько раз робот переспросит клиента, если не получит или не сможет распознать ответ на свой вопрос. Для стабильной работы функции нужно добавить одну или несколько фраз переспроса ниже в блоке Фразы с просьбой повторить ответ.</p> <p>В голосовых ботах по истечении счетчика переспросов звонок будет сброшен или переведен на оператора, в случае если настроен перевод на оператора. В чатах всегда будет совершена попытка перевода на оператора.</p>
Выгрузка всех текстов	<p>Позволяет скачать все фразы из блоков сценария. После нажатия на кнопку файл с фразами в формате .txt будет загружен на компьютер. Может быть полезно при озвучивании робота с помощью аудиозаписей.</p>
<p><i>Фразы с просьбой повторить ответ</i></p>	
Фраза	<p>Укажите фразу, которую робот будет воспроизводить в диалоге с клиентом в случае, если ответ не получен или не удастся его обработать. Каждая фраза должна быть вписана в отдельном поле. Поля для фраз добавляются по кнопке Добавить и удаляется по кнопке .</p>
<p><i>Фразы с просьбой сказать следующую часть большого числа</i></p>	

Фраза	<p>Укажите фразу, которую робот будет воспроизводить в диалоге с клиентом в случае, если произнесенное клиентом число содержит меньше символов, чем ожидалось. Каждая фраза должна быть вписана в отдельном поле. Поля для фраз добавляются по кнопке . Добавить и удаляется по кнопке . Используется для стрелок принятия решений с типом выходных данных Большое число и Номер телефона.</p>
-------	--

Перевод на специалиста

Активируйте переключатель, чтобы в случае, когда робот не знает что делать, выполнялась переадресация на оператора.

Фразы перевода на специалиста	Введите фразы, которые робот будет озвучивать при переводе диалога на оператора.
-------------------------------	--

Тип входящей линии	Выберите действие при переадресации. Доступны следующие:	
	Подключить вызов к каналу	Будет выполнена переадресация на предустановленное направление перевода. В появившемся поле Канал необходимо выбрать нужный канал. Каналы для перевода создаются в разделе Голосовые боты → Настройки → Переводы.
	Произвольное направление перевода	При выборе этого значения будет выполнена переадресация на указанный номер с указанным заголовком. В появившихся полях настройте назначение и заголовки сопровождения.
	Другой сценарий	При выборе этого значения будет выполнен перевод на другого бота. В появившемся поле Бот необходимо выбрать сценарий, на который будет переведен клиент.

Группа операторов (для чата)	С помощью групп операторов можно переводить диалоги с клиентами в разные отделы вашей компании или на разных специалистов, например, в отдел продаж, бухгалтерию, техническую поддержку.
<p><i>Активное слушание</i></p> <p>Это функция голосовых ботов, которая делает диалог более естественным. Она вставляет синтезированные или записанные фразы в ответ на речь пользователя, имитируя поведение живого собеседника. Подробнее об активном слушании читайте в статье Активное слушание.</p> <p>Чтобы активировать функцию активного слушания, переключите переключатель в активное положение и задайте необходимые настройки. Тогда активное слушание будет распространяться на все блоки сценария Вопрос и Пауза.</p>	
Мин. интервал озвучки (мс)	Начало интервала от начала речи пользователя, в течение которого можно запускать фразы озвучки.
Макс. интервал озвучки (мс)	Конец интервала, в течение которого можно запускать фразы озвучки.
Фразы озвучки	<p>Список фраз, которые будут озвучены пользователю в случайное время в указанном выше промежутке. Фразы произносятся в случайном порядке.</p> <p>Фразы добавляются по кнопке Добавить и удаляются по кнопке .</p>
<i>Дополнительно</i>	

<p>Приведение слов к первой форме</p>	<p>Позволяет приводить все слова в ответах клиента в начальную форму для облегчения распознавания речи клиента. Существительные приводятся к форме единственного числа и именительного падежа (например, менеджер), глаголы приводятся в неопределенную форму (например, позвонить), прилагательные приводятся к форме единственного числа, именительного падежа, мужского рода (например, синий). Данная функция также применима при использовании ключевых слов.</p> <p><input type="checkbox"/> Работает только для русского и английского языков.</p>
<p>Фонетический корректор при звонках</p>	<p>Позволяет исправлять мелкие ошибки в произношении слов, например: «пирвет» → «привет».</p>
<p>Блок по умолчанию</p>	<p>Блок в сценарии, в который система направляет робота, если он не получает или не может распознать ответ клиента на свой вопрос. Выбирается из блоков Вопрос в текущем сценарии.</p>
<p>Фоновые шумы</p>	<p>Фоновый шум, который позволяет создать желаемую атмосферу во время диалога с клиентом, например: звуки офиса, улицы, аэропорта.</p>
<p>Таймаут ожидания</p>	<p>Период времени ожидания ответа в секундах, в течение которого робот будет ждать от клиента ответ на вопрос. Если клиент ответит на вопрос робота по истечении указанного времени и его ответ не будет соответствовать заданному вопросу, то система применит его ответ к блоку Вопрос, который указан в качестве блока помощи.</p>
<p>Количество циклов</p>	<p>Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.</p>
<p>Блок помощи вопрос</p>	<p>Блок, к которому система применяет ответ клиента, если клиент не ответил на вопрос в течение указанного времени, и его ответ не подходит под заданный вопрос. Выбирается из блоков Вопрос в текущем сценарии.</p>

Перебивания

Используется для того, чтобы клиент мог прерывать высказывание бота в диалогах, используя определенные стоп-слова. При произнесении стоп-слова во время высказывания бота, его речь завершается, и происходит переход к следующему этапу сценария. В стартовом блоке задаются стоп-слова уровня сценария. Подробнее о перебиваниях читайте в статье [Перебивания](#).

Перебивания платформы	Показывает список стоп-слов уровня платформы.
Перебивания компании	Показывает список стоп-слов уровня компании.
Игнорировать вышестоящие перебивания	Позволяет отключить перебивания уровня платформы и уровня компании.
+Добавить	Добавление стоп-слов по одному.
Перебивания платформы	Показывает список стоп-слов уровня платформы.
Перебивания компании	Показывает список стоп-слов уровня компании.
Игнорировать вышестоящие перебивания	Позволяет отключить перебивания уровня платформы и уровня компании.
+Добавить	Добавление стоп-слов по одному.

<p>+Добавить список перебиваний</p>	<p>Добавление стоп-слов группой. Введите слова, отделяя их запятыми для распознавания системой.</p>			
<p><i>Валидация озвучки</i> Процесс проверки наличия файлов озвучки для различных типов блоков в сценарии при его сохранении. Если в настройках отключить проверку для конкретного блока, то при сохранении сценария система не будет проверять наличие озвучки в этих блоках. Ниже представлены переключатели для включения или отключения проверки озвучки в различных блоках редактора сценария для создания ботов.</p>				
<p><i>Переменные и шаблоны</i></p>				
<p><i>Переменные</i> Дает возможность создать переменные и задать их типы. Каждая переменная добавляется в отдельном поле по кнопке Добавить и удаляется по кнопке  .</p>				
<p>Название</p>	<p>Название переменной. В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.</p>			
<p>Тип</p>	<p>Тип переменной. Доступны следующие типы:</p> <table border="1" data-bbox="357 1547 1356 1973"> <tr> <td data-bbox="357 1547 596 1973"> <p>Имя или отчество</p> </td> <td data-bbox="596 1547 1356 1973"> <p>Значениями этого типа могут быть следующие данные:</p> <ul style="list-style-type: none"> ● Фамилия; ● Имя; ● Отчество; ● Фамилия, имя; ● Фамилия, имя, отчество. </td> </tr> </table>		<p>Имя или отчество</p>	<p>Значениями этого типа могут быть следующие данные:</p> <ul style="list-style-type: none"> ● Фамилия; ● Имя; ● Отчество; ● Фамилия, имя; ● Фамилия, имя, отчество.
<p>Имя или отчество</p>	<p>Значениями этого типа могут быть следующие данные:</p> <ul style="list-style-type: none"> ● Фамилия; ● Имя; ● Отчество; ● Фамилия, имя; ● Фамилия, имя, отчество. 			

Текстовые данные	Текстовые данные длиной более 100 символов.
Дата	Значения даты в формате ГГГГ-ММ-ДД, времени в формате ЧЧ:ММ:СС, а также даты и времени в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС+00:00.
Время	Значения времени в формате ЧЧ:ММ:СС.
Дата и время	Значения даты и времени в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС+00:00.
Сумма в рублях	<p>Целое число, которое робот произносит как сумму в рублях.</p> <p><input type="checkbox"/> Работает только для рублей, другие валюты не поддерживаются.</p>
Аббревиатура	Данные, которые робот произносит по буквам, например: «где» робот произнесет как «г», «д», «е».
Номер телефона	Данные длиной 11 символов в следующем формате: 1 символ (код страны), 3 символа (код города), 3 символа, 2 символа, 2 символа. Эти данные робот произносит по разрядам номера телефона: «Восемь, девятьсот тринадцать, двести сорок шесть, тринадцать, тринадцать».
Большое число	<p>Числа длиной более трех символов. По умолчанию перед синтезом значения разбиваются на группы по 3 символа, например, число 1 982 147 робот произнесет как 198, 214, 7.</p> <p><input type="checkbox"/> Число можно разбить на любую разрядность через форматтер {имя переменной формат:разрядность}.</p>

	Число	Числа длиной не более трех символов.
	Неопределенный	Любые текстовые данные длиной не более 100 символов.
Шаблоны ключевых слов	Позволяет задать ключевые слова на уровне сценария для веток, выходящих из блока Вопрос с типом выходных данных Ключевые слова.	

6.9.2. Информация

Блок **Информация** служит для передачи сообщения клиенту, где бот предоставляет определенную информацию и переходит к следующему блоку без ожидания ответа клиента. Это один из двух основных блоков при создании сценария.

Из блока можно провести стрелку только в один блок.

Свойства блока

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Свойство	Описание
Общее	
Основное	
Подпись	Название блока, которое отображается в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства при создании сценария.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.

Сообщение	Сообщение, которое будет озвучено или отправлено роботом после перехода в блок. Текст сообщения отображается в блоке в редакторе сценария, если не указана Подпись, а также высвечивается в виде всплывающей подсказки при наведении на блок.
Задержка	Период времени, по истечении которого будет отправлено сообщение после перехода в блок. С помощью задержки перед отправкой можно эмулировать набор сообщения, создавая впечатление, что общение ведется с реальным человеком. Задается в секундах.
Количество циклов	Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.
Немедленная отправка	Позволяет отправить сообщения до полного формирования очереди в связке с блоком Запрос к серверу. Например, в сценарии прописана цепочка из 3 блоков Информация, блока Запрос к серверу и еще 2 блоков Информация. Если поставить галочку в поле Немедленная отправка, на третьем блоке в этой цепочке (т.е. перед блоком Запрос к серверу), уйдут 3 сообщения, потом отправится запрос и уйдут оставшиеся 2. Иначе сообщения отправятся только после обработки блока Запрос к серверу.
Файлы	<p>Позволяет добавить файлы, которые будут прикреплены к сообщению бота. Файлы добавляются по кнопке  и отображаются каждый в отдельной плашке. Файлы удаляются по клику на .</p> <p><input type="checkbox"/> Максимальный объем файла — 15 МБ.</p> <p>При тестировании сценария в редакторе сценариев загруженный файл не отобразится; для проверки корректности отображения воспользуйтесь виджетом.</p>

Переменные

Дает возможность создать переменные и задать им значения. Каждая переменная добавляется в отдельном поле по кнопке Добавить и удаляется по кнопке .

Присваивать переменные в блоке Информация не рекомендуется, так как в длинном сценарии с ними будет сложно работать. Лучше присваивать переменные в блоке Результат.

Название

Название переменной.

В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.

Значение

Значение переменной.

Пропуск блока

Система пропустит текущий блок, если она озвучит информационный блок, который будет выбран в данном пункте.

Идентификатор блока

Вписывается ID блока, в случае перехода к которому необходимо пропустить текущий блок. Чтобы узнать ID нужного блока, нажмите на него в сценарии и в открывшихся свойствах блока справа от заголовка Основное нажмите на кнопку ID. Идентификатор будет скопирован, после этого вы сможете вставить его в поле в разделе Пропуск блока.

<p>Фиксация результата</p> <p>Используется для сохранения данных диалога в отчет, в котором будут заполнены соответствующие полям столбцы. Чтобы фиксировать результат, включите переключатель.</p>			
Дата	Дата фиксации для выгрузки в отчет.		
Оценка	Оценка действия для выгрузки в отчет.		
NPS	Произвольная информация (переменная, ответ пользователя и т.д.), которая будет использоваться для анализа диалогов.		
Подтверждение	Краткий смысл происходящего в данном блоке.		
<p>Действия</p> <p>Позволяет выбрать одно из вложенных действий.</p>			
Добавить действие	Доступны следующие типы действий:		
	Действие	Настройки	Описание
	Пользовательское событие Интеграционный функционал для создания пользовательских событий.	Сообщение на кнопке	Текст, который отображается на кнопке клиента.
Название события		Заголовок пользовательского события.	

	Данные	Описание пользовательского события.
Прикрепить шаблон WhatsApp Позволяет прикрепить шаблон для WhatsApp.	Идентификатор шаблона	Уникальный идентификатор полученный от технической поддержки.
	Название шаблона	Название шаблона, полученное от технической поддержки.
	Сообщение на кнопке	Текст, который отображается на кнопке клиента.
	Ссылка на картинку	URL картинки.
	Параметры	Возможность добавить переменную.
	Тип кнопок	Тип кнопок влияет на внешний вид кнопок в мессенджере.
Кнопки WhatsApp Позволяет добавлять кнопки в WhatsApp.	Тип кнопок	Тип кнопок влияет на внешний вид кнопок в мессенджере.

		Header	Текст, который отображается в верхней части сообщения.
		Footer	Текст, который отображается в нижней части сообщения.
		Кнопки	Позволяет добавить нужное количество кнопок и указать для каждой кнопки один из возможных вариантов ответа.

6.9.3. Вопрос

Блок **Вопрос** используется для получения ответа клиента. Этот блок помогает «выслушать» ответ клиента, обработать и направить его с помощью выходящих из блока стрелок. Это один из двух основных блоков при создании сценария.

В отличие от блока **Пауза**, в блоке **Вопрос** ожидание ответа клиента ограничено длительностью распознавания, выставленной в параметрах блока.

Свойства блока

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Свойств о	Описание
Общее	

Основное	
Подпись	Название блока, которое отображается в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства при создании сценария.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Сообщение	Вопрос, который произнесет робот при попадании в этот блок. Текст отображается в блоке в редакторе сценария, если не задана Подпись, а также высвечивается как всплывающая подсказка при наведении на блок.
Задержка	Период времени, по истечении которого будет отправлено сообщение после перехода в блок. С помощью задержки перед отправкой можно эмулировать набор сообщения, создавая впечатление, что общение ведется с реальным человеком. Задается в секундах.
Количество циклов	Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.
Немедленная отправка	Позволяет отправить сообщения до полного формирования очереди в связке с блоком Запрос к серверу. Например, в сценарии прописана цепочка из 3 блоков Информация, блока Запрос к серверу и еще 2 блоков Информация. Если поставить галочку в поле Немедленная отправка на третьем блоке в этой цепочке (т.е. перед блоком Запрос к серверу), уйдут 3 сообщения, потом отправится запрос и уйдут оставшиеся 2. Иначе сообщения отправятся только после обработки блока Запрос к серверу.

Активное слушание

Это функция голосовых ботов, которая делает диалог более естественным. Она вставляет синтезированные или записанные фразы в ответ на речь пользователя, имитируя поведение живого собеседника. Подробнее об активном слушании читайте в статье [Активное слушание](#).

Чтобы настройки активного слушания использовались для указанного блока, нажмите + Добавить активное слушание. Если же вы не хотите применять эту функцию к данному блоку, нажмите переключатель. При удалении активного слушания для блока будут использоваться настройки стартового блока.

Таким образом, разница между отключением и удалением заключается в том, что при отключении активное слушание не используется в конкретном блоке, но его настройки сохраняются, а при удалении настройки блока сбрасываются на настройки стартового блока.

Мин. интервал озвучки (мс)	Начало интервала от начала речи пользователя, в течение которого можно запускать фразы озвучки.
Макс. интервал озвучки (мс)	Конец интервала, в течение которого можно запускать фразы озвучки.
Фразы озвучки	Список фраз, которые будут озвучены пользователю в случайное время в указанном выше промежутке. Фразы произносятся в случайном порядке. Фразы добавляются по кнопке Добавить и удаляются по кнопке  .

Файлы

Позволяет добавить файлы, которые будут прикреплены к сообщению бота.

Максимальный объем файла — 15 МБ.

При тестировании сценария в редакторе сценариев загруженный файл не отобразится; для проверки корректности отображения воспользуйтесь виджетом.

Распознавание

Используется только для голосового бота.

[Настройки распознавания](#) речи клиента.

<p>Длительность</p>	<p>Период времени, в течение которого система выполняет распознавание речи абонента во время вызова.</p> <p>Возможные варианты:</p> <ul style="list-style-type: none"> ● Односложный ответ; ● Очень-очень короткая; ● Очень короткая; ● Короткая; ● Нормальная; ● Нормальная (5сек); ● Нормальная (180сек); ● Длинная; ● Очень длинная; ● Очень длинная (180сек); ● Пользовательская. 							
	<p>Пользовательская</p> <p>Позволяет настроить собственные параметры распознавания.</p> <p><input type="checkbox"/> Полноценная работа этих настроек гарантируется только с системой ASR TWIN. В других системах некоторые из этих настроек могут работать ограниченно или не поддерживаться вовсе.</p>	<table border="1"> <tr> <td data-bbox="935 824 1161 1115"> <p>Пауза между словами</p> </td> <td data-bbox="1161 824 1356 1115"> <p>Укажите значение в диапазоне от 100 мс до 10 000 мс.</p> </td> </tr> <tr> <td data-bbox="935 1115 1161 1406"> <p>Длительность тишины</p> </td> <td data-bbox="1161 1115 1356 1406"> <p>Укажите значение в диапазоне от 100 мс до 10 000 мс.</p> </td> </tr> <tr> <td data-bbox="935 1406 1161 1691"> <p>Общая длительность сессии</p> </td> <td data-bbox="1161 1406 1356 1691"> <p>Укажите значение в диапазоне от 1 000 мс до 180 000 мс.</p> </td> </tr> </table>	<p>Пауза между словами</p>	<p>Укажите значение в диапазоне от 100 мс до 10 000 мс.</p>	<p>Длительность тишины</p>	<p>Укажите значение в диапазоне от 100 мс до 10 000 мс.</p>	<p>Общая длительность сессии</p>	<p>Укажите значение в диапазоне от 1 000 мс до 180 000 мс.</p>
<p>Пауза между словами</p>	<p>Укажите значение в диапазоне от 100 мс до 10 000 мс.</p>							
<p>Длительность тишины</p>	<p>Укажите значение в диапазоне от 100 мс до 10 000 мс.</p>							
<p>Общая длительность сессии</p>	<p>Укажите значение в диапазоне от 1 000 мс до 180 000 мс.</p>							

Количество переспросов	<p>Сколько раз робот будет переспрашивать клиента, если не получит или не сможет распознать ответ на свой вопрос. По достижении указанного количества произойдет перевод на оператора. По умолчанию указано 3.</p> <p><input type="checkbox"/> Для работы этой функции необходимо добавить одну или несколько фраз переспроса в разделе Фразы с просьбой повторить ответ.</p>
Использовать ответ пользователя	<p>Использовать ответ клиента, сохраненный в системной переменной в блоке Вопрос ранее по сценарию. Данную функцию удобно использовать, когда во время диалога осуществляется переход из одного сценария в другой. В этом случае, в первом сценарии происходит сохранение ответа клиента в системную переменную, а во втором сценарии бот использует уже полученный ответ, чтобы не задавать клиенту вопрос, ответ на который был уже получен. После использования сохраненный ранее ответ клиента автоматически удаляется.</p>
Отключить переспрос	<p>Фразы переспроса не будут задействованы, при нераспознанном ответе клиента робот перейдет к ветке по умолчанию. Если ветка по умолчанию не настроена, диалог будет сброшен.</p>
Отключить возврат	<p>Отключить возврат на предыдущий блок Вопрос для поиска вариантов обработки ответов клиента.</p>
<p><i>Настройки клавиатуры Telegram</i></p>	
Кнопка на строку	<p>Максимальное количество кнопок в одной строке в Telegram.</p>
<p><i>Фразы с просьбой повторить ответ</i></p>	

Фраза	<p>Фразы с просьбой повторить ответ, которые озвучиваются роботом по одной, если ответ не получен или не удастся его обработать. Каждая фраза должна быть вписана в отдельном поле.</p> <p>Фразы, указанные в блоке приоритетнее фраз для всего сценария, которые указаны в стартовом блоке. Это значит, что если в данном блоке будут добавлены фразы с просьбой повторить ответ, то бот будет использовать их для переспроса, в противном случае бот будет использовать фразы, заданные в стартовом блоке.</p>
Отключить рандом фраз повтора	<p>Отключить случайный выбор фраз повтора. Если поставить галочку, фразы будут озвучиваться в том порядке, в котором они указаны в разделе, иначе – в случайном порядке.</p>
<p><i>Переменные</i></p> <p>Дает возможность создать переменные и задать им значения. Каждая переменная добавляется в отдельном поле по кнопке Добавить и удаляется по клику на .</p>	
Название	<p>Название переменной.</p> <p>В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.</p>
Значение	<p>Значение переменной.</p>
<p><i>Фиксация результата</i></p> <p>Используется для сохранения данных диалога в отчет, в котором будут заполнены соответствующие полям столбцы. Чтобы фиксировать результат, активируйте переключатель.</p>	
Дата	<p>Дата фиксации для выгрузки в отчет.</p>
Оценка	<p>Оценка действия для выгрузки в отчет.</p>

NPS	Произвольная информация (переменная, ответ пользователя и т.д.), которая будет использоваться для анализа диалогов.
Подтверждение	Краткий смысл происходящего в данном блоке. Например: «Уточняем у клиента цвет авто».
<p><i>Перебивания</i></p> <p>Используется для того, чтобы клиент мог прерывать высказывание бота в диалогах, используя определенные стоп-слова. При произнесении стоп-слова во время высказывания бота его речь завершается, и происходит переход к следующему этапу сценария. В блоке Вопрос задаются стоп-слова уровня блока. Подробнее о перебиваниях читайте в статье Перебивания.</p>	
Перебивания платформы	Показывает список стоп-слов уровня платформы.
Перебивания компании	Показывает список стоп-слов уровня компании.
Перебивания сценария	Показывает список стоп-слов уровня сценария.
Игнорировать вышестоящие перебивания	Позволяет отключить перебивания уровня платформы, уровня компании и уровня сценария.
+Добавить	Добавление стоп-слов по одному.
+Добавить список перебиваний	Добавление стоп-слов группой. Введите слова, отделяя их запятыми для распознавания системой.

Действия

Добавить действие

Позволяет выбрать одно из вложенных действий. Доступны следующие типы действий:

Пользовательское событие
Интеграционный функционал для создания пользовательских событий.

Сообщение на кнопке

Текст, который отображается на кнопке клиента.

Название события

Заголовок пользовательского события.

Данные

Описание пользовательского события.

Принудительный переход к блоку
Позволяет переходить к выбранному блоку аналогично работе [блока Телепорт](#).

Подпись

Произвольное описание.

Идентификатор блока

ID выбранного блока для перехода.

Прикрепить шаблон WhatsApp
Позволяет прикрепить шаблон для WhatsApp.

Идентификатор шаблона

Уникальный ID полученный от технической поддержки.

		Название шаблона	Название шаблона, полученное от технической поддержки.
		Сообщение на кнопке	Текст, который отображается на кнопке клиента.
		Ссылка на картинку	URL картинки.
		Параметры	Возможность добавить переменную.
		Тип кнопок	Тип кнопок влияет на внешний вид кнопок в мессенджере.
	Кнопки WhatsApp Позволяет добавлять кнопки в WhatsApp.	Тип кнопок	Тип кнопок влияет на внешний вид кнопок в мессенджере.
		Header	Текст, который отображается в верхней

			части сообщения.
		Footer	Текст, который отображается в нижней части сообщения.
		Кнопки	Позволяет добавить нужное количество кнопок и указать для каждой кнопки один из возможных вариантов ответа.
Поделиться номером Позволяет получить номер от клиента через кнопку.		Сообщение на кнопке	Текст, который отображается на кнопке клиента.
WebApp приложение Позволяет открывать веб-приложения в Telegram.		Название кнопки	Текст, который отображается на кнопке в Telegram.
		Ссылка на приложение	Ссылка на сайт, который будет открываться

			я при нажатии на кнопку.
	Запрос геопозиции Позволяет запросить географическое положение пользователя. Работает в Telegram и Viber.	Название кнопки	Текст, который отображается на кнопке клиента.
Добавить форму Позволяет прикрепить форму для заполнения данных. Работает в виджете.	Шаблон	Позволяет выбрать преднастроенный шаблон или создать свою форму. Доступны следующие формы: <ul style="list-style-type: none"> ● Запрос имени и email; ● Запрос ФИО; ● Запрос телефона; ● Запрос выбора ; ● Создать свою. 	

Стрелки принятия решений

Свойства стрелок

Чтобы перейти к свойствам, нажмите на стрелку. Справа откроется окно со свойствами.

Свойство	Описание
Основное	
Подпись	<p>Название стрелки в сценарии. По названию можно быстро найти нужную стрелку и понять ее назначение.</p> <p>Поле Подпись появится только после выбора типа выходных данных.</p> <p>Если для стрелки не задана подпись, вместо нее в качестве названия стрелки будет отображаться тип выходных данных. Пока тип также не будет задан, стрелка будет называться Пустая.</p>
Тип выходных данных	<p>Определяет тип условия, при выполнении которого система перейдет по направлению, заданному данной стрелкой.</p> <p>Новые поля для заполнения отобразятся в зависимости от выбранного в стрелке типа выходных данных. Типы выходных данных описаны в таблице ниже.</p>
Сохранить ответ	<p>Настраивается, чтобы сохранить ответ клиента в переменную <code>{answer}</code>, что позволит в дальнейшем использовать его в блоках сценария.</p>
Название переменной	<p>Название переменной, в которую будет сохранен текущий ответ клиента.</p> <p>В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.</p>

Сохранить всю фразу	Определяет способ сохранения ответа. Если включено, система сохранит ответ клиента полностью. Если не включено, система сохранит только ключевое слово, по которому осуществляется переход в ветку.
Сделать переменную системной	Сохранить ответ пользователя в переменную таким образом, чтобы в дальнейшем использовать ее в других сценариях. Полезно при переходе диалога из одного сценария в другой, так как позволяет не задавать пользователю вопрос, ответ на который был получен ранее.

Типы выходных данных

Этот параметр должен быть заполнен для каждой стрелки принятия решения. Он определяет тип условия, при выполнении которого система понимает, что диалог должен развиваться в направлении, которое указано с помощью этой стрелки.

Тип выходных данных	Описание
По умолчанию	<p>Переход по ветке будет произведен в случае, если условия во всех остальных ветках неверны.</p> <p><input type="checkbox"/> Всегда добавляйте к блоку Вопрос ветку по умолчанию, так как все действия клиента невозможно предугадать, а данная стрелка обеспечит развитие сценария в случае, если ответ клиента не был запланирован в других ветках сценария.</p>
Быстрый ответ	Вариант автоматического ответа, название кнопки ответа.
Номер	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>

	Вес ветки	<p>Количественный вес ветки, ее приоритет среди остальных веток.</p> <p>Если существует несколько веток по умолчанию, будет выбрана та, что в данный момент по весу подходит больше всего, в зависимости от завершенных на данный момент циклов переспроса. Позволяет настраивать выбор ветки по умолчанию при множественных переспросах.</p>
Ключевые слова	Переход по ветке будет произведен, если ответ пользователя совпадет с ключевым словом.	
	Ключевые слова	<p>Ключевые слова, с которыми должен совпасть ответ пользователя.</p> <p>Правила добавления ключевых слов:</p> <ol style="list-style-type: none"> 1. Ключевые слова указываются через пробел. 2. Выражения из нескольких слов указываются в кавычках, например: «уже на месте». 3. Ключевым словам можно добавить вес с помощью знаков + (повысить вес) и - (понижить вес), где каждый знак имеет единицу веса (слово или выражение). <p>Например, ответ «да» в контексте не всегда может означать согласие. Во фразе «Да, я вас отчетливо слышу, я не пойду на обед» частица «не» приоритетнее ответа «да», и ее вес должен быть больше. При расстановке ключевых слов и выражений можно добавить вес к определенно отрицательным словам: +++не и +нет – тогда будет считаться, что к слову нет добавлено одно слово, а к частице не – 3 слова. При количественном перевесе система принятия решения выберет соответствующий маршрут.</p> <p><input type="checkbox"/> По умолчанию, вес фразы выше веса отдельного слова и равен количеству слов во фразе.</p>
	Шаблоны ключевых слов	<p>Возможность использования заранее подготовленного набора ключевых слов. Шаблоны задаются в настройках стартового блока на вкладке Переменные и шаблоны.</p>

	Быстрый ответ	Вариант автоматического ответа, название кнопки ответа. <input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.
	Номер	Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами. <input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.
Дата/Время/Дата и время	Для перехода по ветке ответ клиента должен содержать дату, время или дату и время (для каждого – свой тип). Например, один из данных типов используется, когда робот спрашивает у клиента дату его рождения или дату и время записи на прием к врачу.	

Значение даты/времени/даты и времени

Задается логическое условие (как правило, оператор сравнения), при выполнении которого робот пойдет по выбранной стрелке и, в зависимости от условия, далее в появившихся полях необходимо задать одно значение, два значения или не указывать ничего.

- Значение может быть равно, меньше, меньше или равно, больше, больше или равно в сравнении со значением, указанным клиентом. При выборе одного из соответствующих условий нужно задать значение в появившемся одноименном поле: Значение даты / Значение времени / Значение даты и времени.
- Значение может находиться в некотором периоде, в диапазоне между двумя значениями. Выбирается условие Между, в появившихся полях Значение даты, от / Значение времени, от / Значение даты и времени, от и Значение даты, до / Значение времени, до / Значение даты и времени, до задаются даты / время / даты и время начала и конца периода. При этом значение во втором поле не может быть меньше или равно значения в первом поле.
- Могут засчитываться любые дата / время / дата и время, названные клиентом – в этом случае главное, чтобы робот прошел по стрелке, если клиент назвал дату / время / дату и время. Для этого выбирается вариант Любой, дополнительные поля не заполняются.

Например, робот хочет проверить, есть ли пользователю 18 лет, и просит его ввести дату рождения – в этом случае задается Значение даты = Меньше и дата на 18 лет раньше текущей.

□ Значение даты вводится в формате YYYY-MM-DD. Значение времени вводится в формате HH:mm (от 00:00 до 23:59).

□ Значение даты и времени вводится в формате YYYY-MM-DD HH:mm (время – от 00:00 до 23:59).

	Быстрый ответ	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p><input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.</p>
	Номер	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>
Число	<p>Переход по ветке будет произведен, если озвученное пользователем число удовлетворяет условиям в свойствах стрелки.</p>	
	Значение числа	<p>Необходимо задать условие, при выполнении которого робот пойдет по выбранной стрелке, и в зависимости от условия задать число, числа или не указывать ничего.</p> <ul style="list-style-type: none"> • Число может быть равно, меньше, больше или равно, больше, больше или равно в сравнении с числом, указанной клиентом. Выбирается одно из соответствующих условий, число задается в появившемся одноименном поле Значение числа. • Число может находиться в диапазоне между двумя числами. Выбирается условие Между, в появившихся полях Значение числа, от и Значение числа, до задаются даты начала и конца периода. При этом значение во втором поле не может быть меньше или равно значения в первом поле. • Может засчитываться любое число, названное клиентом – система будет реагировать на любые цифры. Для этого выбирается вариант Любой, дополнительные поля не заполняются. <p>Например, робот спрашивает клиентку магазина косметики: «Подскажите, сколько вам полных лет?». Если клиентка называет больше 40, ей будет предложен антивозрастной крем. Для этого нужно задать Значение даты = Больше и значение даты = 40.</p>

	Быстрый ответ	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p>На каждый быстрый ответ добавляется своя стрелка.</p>
	Номер	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами (первая, вторая, третья и т. д.).</p> <p>□ Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>
Большое число	Значение числа	<p>Переход по ветке будет произведен, если озвученное пользователем число удовлетворяет условиям в свойствах стрелки. Система будет ожидать число длиной более 3-х символов.</p> <p>Необходимо задать условие равенства определенному числу, при выполнении которого робот пойдет по выбранной стрелке, или указать, что для перехода по стрелке будет приниматься любое число.</p> <ul style="list-style-type: none"> • Если указанное число должно быть равно числу, введенному клиентом – выбирается вариант Равно, нужное число задается в появившемся одноименном поле Значение числа. <p>□ Прочие операторы сравнения для типа Большое число недоступны, но их можно использовать с типом Число.</p> <ul style="list-style-type: none"> • Может засчитываться любое число, названное клиентом – система будет реагировать на любые цифры. Для этого выбирается вариант Любой, дополнительные поля не заполняются. <p>Например, робот может уточнить у клиента серию и номер паспорта и проверить, соответствуют ли (равны ли) они его данным, чтобы продолжать разговор с клиентом.</p>

	<p>Маска числа</p>	<p>Маска ожидаемого ввода, определяет количество цифр. Должна состоять исключительно из символов #. Например, для паспорта задается маска вида #####, где каждый символ # – это цифра.</p>
	<p>Одна попытка</p>	<p>Пользователю дается только одна попытка, чтобы ввести число. Например, если робот подтверждает клиента по серии и номеру паспорта, клиент должен ввести их с первой попытки.</p>
	<p>Быстрый ответ</p>	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p><input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.</p>
	<p>Номер</p>	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>
<p>Ошибка определения большого числа</p>	<p>Переход по ветке будет произведен, если в процессе ввода большого числа от клиента было получено что-то не содержащее чисел.</p>	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p><input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.</p> <p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>
<p>Файл</p>		<p>Позволяет системе определить, что клиент отправил файл, реагировать на это, а также получить ссылку на отправленный файл.</p>

NLU	Переход по ветке будет произведен, если указанное в свойствах намерение совпадет с озвученным намерением клиента.	
Агент	Необходимо выбрать обученного агента NLU. Подробнее о NLU читайте в разделе NLU .	
Приоритет ветки	Возможность выставить более высокий приоритет для выбранной ветки относительно ветки с ключевым словом.	
Намерения	<p>Намерения, которые система будет ожидать от пользователя. Каждое намерение выбирается в списке, который открывается по кнопке-плюсу. В списке намерений выбирается конкретное намерение из ранее настроенных в агенте. Также можно задать Любое намерение для выбора любого из намерений указанного агента и Не распознано для перехода к ветке по умолчанию, если намерение не было распознано.</p> <p>Для уже выбранного намерения отображается его название, также вы можете настроить порог доверия в процентах — целое число от 0 до 100. Намерение можно удалить по кнопке с минусом.</p> <p><input type="checkbox"/> В одну стрелку можно добавить несколько намерений – главное, чтобы они были схожи по необходимому посылу. Например, намерения «Перезвоните мне», и «Я занят» могут вести по одной ветке, но при этом нельзя объединять намерения «Дела хорошо» и «Дела плохо». Также при добавлении нескольких намерений нельзя указывать и конкретное намерение, и любое намерение. Для бота это будет означать, что любой ответ можно отнести к данной ветке.</p> <p>Для быстроты обработки ответа рекомендуется ограничиваться максимум 5 намерениями.</p>	

	Сущности	Сущности-переменные, в которые могут быть записаны данные, которые будут выяснены системой во время диалога. Каждая сущность выбирается в списке, который открывается по кнопке-плюсу. В списке сущностей выбирается конкретная сущность из ранее настроенных в агенте. Далее в поле Переменная указывается переменная, в которую будут записаны данные. Если установить флажок Обязательность сущности, то робот перейдет по стрелке только в том случае, если эта сущность будет содержаться во фразе клиента. Сущность можно удалить по кнопке с минусом.	
	Быстрый ответ	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p><input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.</p>	
	Номер	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>	
Сигнал DTMF	<p>Тип сигнала, используемый в телефонии для передачи данных при нажатии клавиш с цифрами на телефоне. Каждая кнопка генерирует уникальный звук, состоящий из двух различных частот. Эта комбинация частот позволяет точно идентифицировать нажатую клавишу.</p> <p>Используется только в голосовых ботах, так как для переходов по веткам используются звуковые сигналы.</p>		
	Тип сигнала	Статически й	<p>Тип сигнала, который фиксирован и не меняется. Пользователь должен ввести заранее заданную комбинацию клавиш. То есть, при настройке сценария указывается конкретный DTMF-сигнал, который ожидается от пользователя.</p>

	Динамический	Тип сигнала, который позволяет пользователю вводить произвольную последовательность цифр и символов, такие как «*» или «#», до определенной длины. Пользователь может ввести любую комбинацию символов, и система примет ее как допустимую, если она не превышает установленную максимальную длину.
	DTMF сигнал	Используется для указания конкретной фиксированной комбинации клавиш, которую должен ввести пользователь на своем телефоне для выполнения определенного действия или перехода к следующему шагу в сценарии. Это поле актуально только для типа DTMF-сигнала Статический. Доступные значения: <ul style="list-style-type: none"> • цифры от 0 до 9; • символы «*» и «#». Эта функция особенно полезна для голосовых меню. Например, бот может сказать клиенту: «Для создания нового заказа нажмите кнопку 1. Для работы с текущим заказом нажмите кнопку 2. Для завершения разговора нажмите кнопку 3».
	Максимальная длина DTMF сигнала	Используется для указания максимального количества символов, которые пользователь может ввести. Это поле определяет верхний предел длины произвольного DTMF сигнала, который система будет принимать и обрабатывать. Поле используется только при типе сигнала Динамический.

<p>Номер телефона</p>	<p>Переход по ветке будет произведен, если количество озвученных символов будет соответствовать ожидаемому количеству цифр в номере телефона.</p> <ul style="list-style-type: none"> ● Параметр Номер телефона позволяет отслеживать как мобильные, так и городские номера. На текущий момент данная функция работает только с номерами российских операторов связи. ● Номер телефона должен содержать не более 11 цифр. В противном случае робот не сможет его распознать. ● При вводе мобильного номера код страны указывать необязательно. ● При вводе городского номера нужно указывать код страны и код города (например, 7 и 812). 	
	<p>Быстрый ответ</p>	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p><input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.</p>
	<p>Номер</p>	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>
<p>Ошибка определена номера телефона</p>	<p>Переход по ветке будет произведен, если озвученный клиентом номер телефона будет недействующим.</p>	
	<p>Быстрый ответ</p>	<p>Вариант автоматического ответа, название кнопки ответа.</p> <p><input type="checkbox"/> На каждый быстрый ответ добавляется своя стрелка.</p>
	<p>Номер</p>	<p>Порядковый номер ответа в списке вариантов ответа, позволяет определить порядок отображения кнопок с ответами.</p> <p><input type="checkbox"/> Если не указать порядок отображения кнопок, система разместит их в произвольном порядке.</p>

6.9.4. Результат

Блок **Результат** служит для фиксации результата разговора, а также для принудительного перевода на оператора, отправки сообщения и отправки нотификации.

Из блока можно провести стрелку только в один блок.

Свойства блока

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Свойство	Описание
Основное	
Подпись	Название блока, которое отображается в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства при создании сценария.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.
Тип действия	Действие, которое будет производить система. Доступные типы действий описаны в таблице ниже.
<i>Переменные</i> Дает возможность создавать переменные и задавать им значения. Каждая переменная добавляется в отдельном поле по кнопке Добавить и удаляется по клику на  .	

Название	<p>Название переменной.</p> <p>В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.</p>
Значение	Значение переменной.
<p><i>Фиксация результата</i></p> <p>Используется для сохранения данных диалога в отчет, в котором будут заполнены соответствующие полям столбцы. Чтобы фиксировать результат, активируйте переключатель.</p>	
Статус	<p>Поле выбора статуса диалога.</p> <p>Нажмите +Добавить статус, чтобы создать новый статус. В поле Название укажите произвольное название для статуса, а в поле Код — код статуса, который будет отображаться в соответствующем столбце отчета.</p> <p>Нажмите на  в строке статуса, чтобы открыть окно редактирования, в котором вы можете изменить или удалить существующий статус.</p>
Дата	Дата фиксации для выгрузки в отчет. В отчетах выводится в столбец Подтверждение.
Оценка	Оценка действия для выгрузки в отчет. В отчетах выводится в столбец Подтверждение.
NPS	Произвольная информация (переменная, ответ пользователя и т.д.), которая будет использоваться для анализа диалогов.
Подтверждение	Поле, в которое можно записать данные для выгрузки в отчет.

Типы блока

Тип	Описание	
Без действия	Робот не совершит никаких действий и перейдет к следующему блоку, при его наличии.	
Произвольный результат	Используется для сохранения собственных переменных и вывода их в отчет. При выборе отобразится раздел Переменные результата, где в полях Название и Значение указываются название и значение переменных. Добавляются по кнопке Добавить, а удаляются по клику на  . Может быть добавлено любое количество полей.	
Отправка email	Используется для отправки электронных писем и предназначен исключительно для внутреннего пользования, например, для отладки и личных уведомлений.	
	Email получателя	Электронная почта, на которую должно прийти письмо. ❑ В email получателя нельзя использовать переменные.
	Заголовок письма	Тема письма. Можно использовать переменные.
	Сообщение	Тело письма. Можно использовать переменные.
	Отправлять транскрипцию	Включает текст диалога робота и клиента в тело письма.
Отправка уведомления	Будет отправлено сообщение по выбранным каналам отправки.	
	Шаблон уведомления	Выпадающий список с заранее настроенными шаблонами. Следующие поля появляются только после выбора шаблона.

	Телефон получателя	Номер телефона получателя сообщения. Номер может быть любым – необязательно клиента, с которым робот в данный момент общается.	
	Email получателя	Адрес электронной почты получателя сообщения. Email может быть любым – необязательно клиента, с которым робот в данный момент общается.	
	Сокращать ссылки	Сокращает длинные ссылки в тексте сообщения с помощью встроенного сервиса.	
	Переменные уведомления	Переменные, использованные в шаблоне (в шаблоне будет прописано название переменной). Каждая переменная добавляется в отдельном поле по кнопке Добавить и удаляется по клику на  . В полях Название и Значение указываются названия и значения переменных.	
	Канал отправки	Каким способом отправить уведомление. Список каналов берется из выбранного шаблона уведомления. Нужный канал отмечается галочкой слева от канала. Если указана отправка по нескольким каналам, то отправка сообщения будет продолжаться до первой успешной отправки в любой канал.	
	Отправить обязательно	Активируйте переключатель, чтобы выполнить отставку в указанный канал в любом случае, даже если успешная отправка уже была выполнена в другом канале.	
Перевод на оператора	Действие при переадресации	Выберите действие при переадресации. Доступны следующие:	
		Подключить вызов к каналу	Будет выполнена переадресация на предустановленное направление перевода. В появившемся поле Канал необходимо выбрать нужный канал. Каналы для перевода

		создаются в разделе Голосовые боты → Настройки→ Переводы.
	Произвольное направление перевода	При выборе этого значения будет выполнена переадресация на указанный номер с указанным заголовком. В появившихся полях настройте назначение и заголовки сопровождения.
	Назначение	Цель или назначение звонков. Например, Продажи.
Заголовки сопровождения	<p>JSON-объект `HelpResultNode` с необходимыми настройками. Эти заголовки будут использоваться вместе с переадресованными звонками.</p> <p>Например, если значение поля Назначение — Продажи, то ваш JSON-объект может выглядеть так:</p> <pre> { "type": "Graph\v1\Node\HelpResultNode", "properties": { "id": { "type": "Graph\v1\Node\NodeId", "properties": { "value": "cbdd7316-db04-4291-8207-31ac536ed93b" } }, "links": [{ "type": "Graph\v1\Link\SimpleLink", "properties": { "id": { "type": "Graph\v1\Node\NodeId", "properties": { "value": "e3a82500-8ed0-45dd-bca6-dbcfea0cc0f9" } }, "comment": "" } }], // Значение взято из блока «Результат» -> «Звонки» -> «Назначение» "destination": "Продажи" } } </pre>	

	Анонс для оператора	<p>Введите сообщение для оператора.</p> <ul style="list-style-type: none"> • В чатах текст анонса виден клиенту и оператору при переводе диалога на оператора. • В телефонии текст анонса проговаривается оператору перед соединением с клиентом. <p>При использовании Произвольного направления перевода телефония должна отдать 200 код в момент соединения с оператором, чтобы он услышал анонс.</p>	
Группа операторов	<p>Функция позволяет выбрать группу операторов, на которую необходимо совершить перевод. С помощью групп операторов можно переводить диалоги с клиентами в разные отделы вашей компании или на разных специалистов. Например, в отдел продаж, бухгалтерию, техническую поддержку.</p> <p><input type="checkbox"/> Работает только для чатов.</p>		
Перезвонить	<p>Робот запланирует повторный звонок данному клиенту в текущем задании на обзвон на определенное время.</p> <p><input type="checkbox"/> Функция не работает при включенном Ф3-230.</p>		
	Тип даты для перезвона	Дата	Указывается в формате: ГГГГ-ММ-ДД чч:мм / ГГГГ-ММ-ДД чч:мм:сс.
		Переменная	Указывается в формате: {переменная}.
Отмена перезвона	<p>Отменяет перезвон, который был запланирован через блок Результат ранее в сценарии.</p>		
Зафиксировать эффективность	<p>Используется для перезвонов в телефонии CIS по правилу результативности.</p> <p><input type="checkbox"/> Настройка Зафиксировать эффективность в блоке Результат приоритетнее настройки результативности вызова в задании на обзвон.</p>		

	Значение эффективности	Не изменять	Робот будет перезванивать согласно настройкам перезвона, указанным в задании на обзвон.
		Результативный	Робот не будет перезванивать, даже если вызов был короче результативности, которая указана в задании на обзвон.
		Нерезультативный	Робот выполнит перезвон клиенту, даже если вызов был длиннее результативности, указанной в задании на обзвон.
Идентифицировать клиента	Настройка для омниканальности, отправка идентификатора клиента выбранному получателю. Дает возможность отправлять сообщения в мессенджер по номеру телефона и продолжать диалог в другом канале.		
	Телефон получателя	Номер телефона получателя идентификатора.	
	Email получателя	Адрес электронной почты получателя идентификатора.	
	Идентификатор клиента	Externalld — это внешний идентификатор клиента, уникальный номер, который помогает системе идентифицировать клиента. Этот номер используется, чтобы не путать объекты между собой, особенно когда системы общаются друг с другом.	
Перевод на другого бота	Перевести клиента на другой сценарий. При выборе отобразится поле Бот, в котором из выпадающего списка сценариев нужно выбрать другой сценарий, по которому продолжится диалог с клиентом.		

6.9.5. Пустой

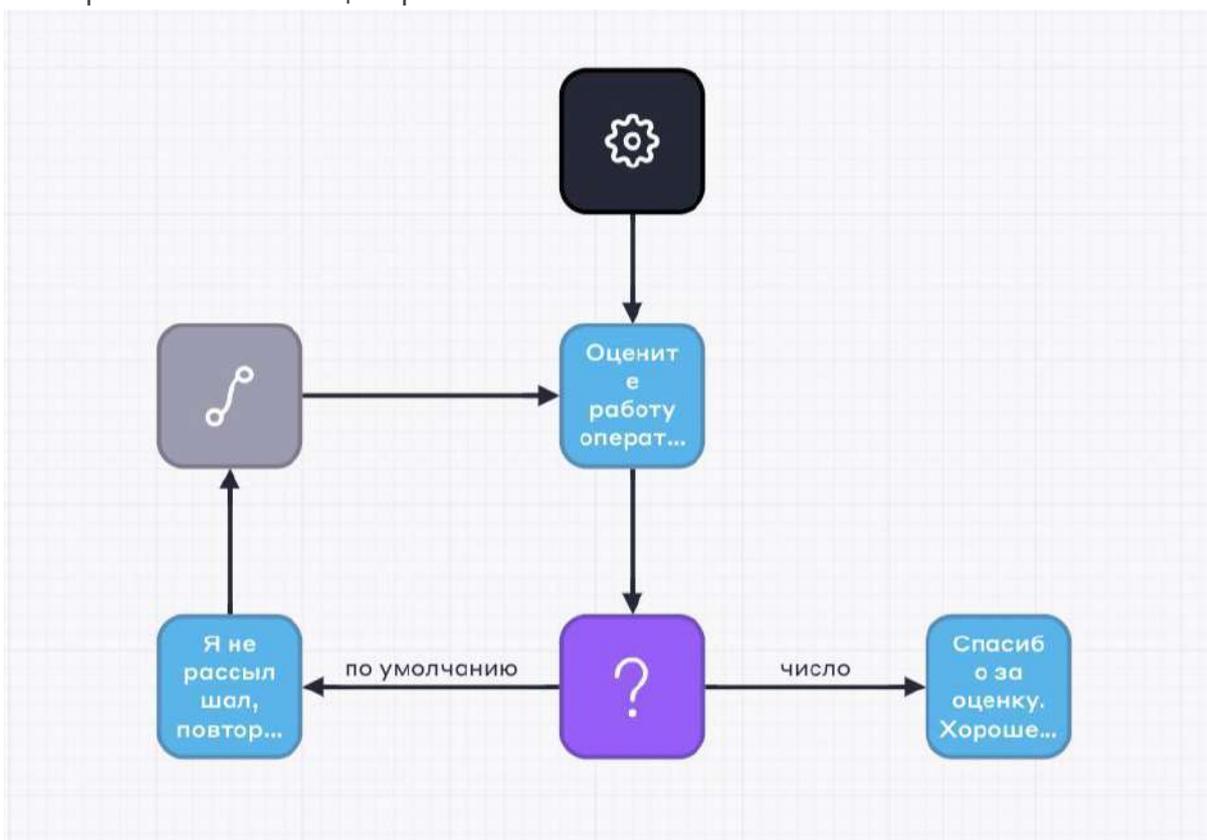
Блок **Пустой** предназначен для удобного формирования структуры сценария. С его помощью можно создавать более организованную структуру блоков и направлять несколько стрелок в один блок. Блок не вызывает реакции системы и служит всего

лишь элементом организации визуального представления сценария, подобно функции стрелки.

Например, вам необходимо разработать сценарий для опроса клиентов и фиксации их ответов. При построении такого сценария возникают трудности, связанные с повторным использованием одних и тех же блоков и стрелок.

Для решения данной проблемы оставьте блок **Вопрос** пустым, а сам вопрос вынесите в блок **Информация**. Робот с помощью блока **Вопрос** слушает ответ клиента и определяет по какой ветке пойти дальше. Чтобы иметь возможность вернуться в блок от которого временно отошли, используйте блок **Пустой**.

Такая конструкция позволяет эффективно управлять вариантами возврата к определенному блоку, избегая необходимости создания новых блоков **Вопрос** и повторного описания сценария.



Из пустого блока можно провести стрелку только в один блок.

Свойство	Описание

Подпись	Название блока, которое отображается в блоке в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства создающего сценарий пользователя.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.

6.7.6. Телепорт

Блок **Телепорт** позволяет перемещаться между разными блоками, словно создавая мост между ними. Этот блок служит для соединения других блоков, особенно в сценариях, где они находятся далеко друг от друга.

Из блока может выходить только одна стрелка.

Свойства блока

Свойство	Описание
Основное	
Подпись	Название блока, которое отображается в блоке в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства создающего сценарий пользователя.

Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое бот может пройти по циклу, после бот переведет на оператора. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.
<i>Отправить в блок</i>	
Блок	Блок, в который будет осуществляться перемещение. Выберите блок из выпадающего списка или введите Название/Сообщение нужного блока в поле поиска.

6.9.7. Порядковый выбор

Блок **Порядковый выбор** используется для последовательного перехода по исходящим из него веткам диалога в рамках одной сессии. То есть, при первом попадании в этот блок произойдет переход по ветке с порядковым номером «1», при повторном попадании в этот блок произойдет переход по ветке с порядковым номером «2», при третьем – по ветке с порядковым номером «3» и т.д.

Из блока можно провести стрелки в любое количество блоков.

Свойства блока

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Свойство	Описание
Подпись	Название блока, которое отображается в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства при создании сценария.

Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.

Свойства стрелки

Свойство	Описание
Подпись	Поле не редактируется и соответствует введенному номеру в поле Номер.
Номер	<p>Задается порядковый номер посещения блока, на основании которого будет выбрана дальнейшая ветка диалога. Может быть введено только число.</p> <p><input type="checkbox"/> Нумерация стрелок должна начинаться с «1» и производиться по порядку. Если не будет доступной стрелки с порядковым номером текущего посещения блока, то будет произведен перевод на оператора.</p>

6.9.8. Случайный выбор

Блок **Случайный выбор** обеспечивает вариативность диалога – система в случайном порядке выполняет переход по стрелкам, которые выходят из блока. Альтернативой к этому блоку является **Порядковый выбор**.

Особые настройки для стрелок, выходящих из блока, не требуются – если в блоке **Порядковый выбор** можно указать номера стрелок, то в этом случае выбор стрелки будет полностью случайным.

Из блока можно провести стрелки в любое количество блоков.

Свойство	Описание
----------	----------

Подпись	Название блока, которое отображается в блоке в редакторе сценария, но не используется в диалоге с клиентом.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое бот может пройти по циклу, после бот переведет на оператора. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.

6.9.9. Запрос к серверу

Блок **Запрос к серверу** позволяет боту общаться с внешней или внутренней системой через API. Робот может искать/получать информацию о клиенте и сохранять новые данные, полученные в диалоге. Также блок позволяет использовать сетевые сервисы через API, такие, как калькулятор, конвертер величин, прогноз погоды и другие. Описание методов API смотрите на [странице](#).

Например, от клиента поступает входящий вызов. Робот определяет номер и обращается к CRM-системе для получения данных. Узнав, что у клиента есть активный заказ, спрашивает, не интересуется ли его статус. При положительном ответе предоставляет информацию о заказе до того, как клиент озвучит свою цель звонка.

Блок не может быть последним в сценарии, должна быть хотя бы стрелка в пустой блок.

Свойства блока

Свойство	Описание
Основное	

Подпись	Название блока, которое отображается в блоке в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства создающего сценарий пользователя.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Сообщение	Текст, который будет воспроизведен роботом после перехода в блок. Отображается в блоке в редакторе сценария, если не задана Подпись, а также высвечивается как всплывающая подсказка при наведении на блок.
Задержка	Период времени, по истечении которого будет отправлено сообщение после перехода в блок. С помощью задержки перед отправкой можно эмулировать набор сообщения – как будто с клиентом общается не бот, а реальный человек. Задается в секундах.
Немедленная отправка	Позволяет отправить сообщения до полного формирования очереди в связке с блоком запроса к серверу.
Как часто говорить еще подождать	Время в секундах, через которое робот будет воспроизводить повторное сообщение до тех пор, пока не будет получен ответ от сервера. Сообщение заполняется ниже в поле Сообщение еще подождать.
Сообщение еще подождать	Текст с просьбой подождать, который будет воспроизведен роботом. Данный текст можно использовать, чтобы заполнить паузу, во время которой система выполняет обращение к API стороннего сервиса или CRM-системы.
<i>Запрос</i>	

Метод	<p>Способ, как клиент (например, веб-браузер) взаимодействует с сервером.</p> <ul style="list-style-type: none"> ● GET — используется для получения данных от сервера. С помощью этого типа запроса можно получать данные о заказе, товаре, пользователе и т.д. ● POST — применяется для отправки новых данных на сервер. С помощью этого типа запроса можно сохранять данные о заказе, создавать товар и т.д. ● DELETE — применяется для удаления данных на сервере. В запросе требуется указать ID изменяемого ресурса. С помощью этого типа запроса можно удалять данные о заказе, товаре и т.д. ● PUT — применяется для внесения изменений в уже имеющуюся на сервере информацию. Используется для полной замены или создания ресурса. Если ресурс существует, то он будет полностью заменен переданными данными. Если ресурс не существует, то он будет создан. С помощью этого типа запроса можно обновлять данные пользователя, редактировать заказ и т.д. ● PATCH — применяется для внесения изменений в уже имеющуюся на сервере информацию. Используется для частичного обновления существующего ресурса, то есть изменения отдельных полей или свойств ресурса. С помощью этого типа запроса можно обновить номер телефона клиента, изменить статус заказа и т.д.
URL	<p>Адрес страницы, на который отправляется запрос, представляет собой основной параметр запроса. Этот адрес привязан к странице, с которой взаимодействует робот, и позволяет использовать переменные в строке адреса для передачи информации в сервис, обеспечивая эффективное взаимодействие между роботом и сервисом.</p>
<i>Парметры</i>	
Ключ	<p>Строка, которая идентифицирует конкретный параметр. Ключ обычно является именем или меткой, которая описывает, что именно передается в запросе. Для добавления ключа нажмите Добавить, для удаления нажмите .</p>

Значение	Конкретное значение, связанное с ключом. Значение представляет собой данные, которые передаются в запросе и соответствуют указанному ключу.	
<i>Авторизация</i>		
Тип авторизации	Тип авторизации в запросе к серверу: <ul style="list-style-type: none"> • Отсутствует; • Логин и пароль; • Заголовок с токеном. Набор следующих заполняемых полей определяется после указания типа авторизации.	
	Отсутствует	Используется, если для работы со страницей не требуется проходить авторизацию. Например, если сервис находится в защищенном контуре.
	Логин и пароль	Используется, если для работы со страницей требуется выполнить авторизацию по логину и паролю.
	Заголовок с токеном	Используется, если для работы со страницей требуется использовать авторизационный токен. Тип токена – схема аутентификации: BASIC, DIGEST, BEARER, MUTUAL, NOVA, AWS4-HMAC-SHA256. Подробнее читайте в статье HTTP-аутентификация . Значение токена – значение ключа. Например, для <code>Authorization: Bearer t490da279fd42889f56</code> в поле Тип токена указывается <code>Bearer</code> , в поле Значение токена указывается <code>t490da279fd42889f56</code> .
<i>Заголовки</i>		
<p><input type="checkbox"/> Куки являются одним из видов заголовков. Если необходимо передать куки, используйте заголовок с именем <code>Cookie</code> и соответствующим значением.</p>		

	Ключ	Название определенного аспекта или свойства запроса. Для добавления ключа нажмите Добавить и удаляется по клику на  .
	Значение	Конкретное значение, связанное с указанным ключом.

Свойства стрелки

Свойство	Описание	
Подпись	Название стрелки, которое отображается над стрелкой в редакторе сценария.	
Тип выходных данных	<p>Тип принятия решения. Переход к следующему блоку сценария осуществляется при достижении определенного результата запроса (успешен или не успешен) или по умолчанию. Следующие поля отобразятся в зависимости от выбранного типа.</p> <p>Если для стрелки не задана подпись и она является стрелкой по умолчанию, в качестве названия стрелки будет отображаться «По умолчанию».</p>	
	По умолчанию	Данный тип используется для перехода по ветке в случае, если условия во всех остальных стрелках данного блока не будут соблюдены. Настройка дополнительных параметров не требуется.
	Успешный ответ	Переход по ветке будет произведен, если ответ от сервера является успешным.
	Код ответа	Код успешного ответа, который система будет ожидать после выполнения запроса. Например, 200, 201. Все коды ответа считаются успешными, даже 500 и 404. Ответ считается неуспешным, если вообще не был получен.

		Условия	<p>Набор условий, по которым принимается решение.</p> <p>Переход по ветке будет произведен, если код ответа от сервера совпадет с указанным в свойствах стрелки и если указанные условия будут выполнены. Результат ответа от сервера будет сохранен в переменную <code>body</code>. Каждое условие добавляется в отдельном поле по кнопке Добавить и удаляется по клику на .</p> <p>Для условий заполняются поля:</p> <ol style="list-style-type: none"> 1. Название – название условия, по которому будет приниматься решение. 2. Оператор – логический оператор в условии: <ul style="list-style-type: none"> ● = – равно, ● != – не равно, ● < – меньше, ● <= – меньше или равно, ● > – больше, ● >= – больше или равно, ● * – содержит, ● ** – совпадает. 3. Значение – значение в условии.
		Переменные	<p>Каждая переменная добавляется в отдельном поле по кнопке Добавить и удаляется по клику на .</p> <p>В полях Название и Поле из ответа указываются названия и значения переменных.</p> <p>В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.</p>

			<p>Для извлечения данных из ответа сервера используйте поле <code>body</code>, например, <code>body.token</code>. Подробнее ознакомиться с примерами вы можете в статье Как получить данные из ответа от сервера.</p>
	Ошибка	<p>Переход по ветке будет произведен, если ответ от сервера получить не удалось. Настройка дополнительных параметров не требуется.</p>	

6.9.10. Условие

Блок **Условие** позволяет задавать условия, которые будут определять поведение робота, и определить направление сценария на основании состояния диалога. Используется для создания сложных цепочек принятия решений.

Из блока можно провести стрелки в любое количество блоков.

Свойства блока

Свойство	Описание
Основное	
Подпись	<p>Название блока, которое отображается в блоке в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства создающего сценарий пользователя.</p>

Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое бот может пройти по циклу, после бот переведет на оператора. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.

Свойства стрелки

Свойство	Описание	
Основное		
Тип выходных данных	Определяет способ принятия решения. Переход к следующему блоку сценария осуществляется при выполнении условия (И / ИЛИ) или по умолчанию. Следующие поля отобразятся в зависимости от выбранного типа.	
	По умолчанию	Используется для перехода по ветке в случае, если условия во всех остальных стрелках данного блока не будут соблюдены.
	Условие И / Условие ИЛИ	<p>Переход по ветке будет произведен, если заданное условие будет выполнено.</p> <ul style="list-style-type: none"> • Условие «И» требует, чтобы все указанные условия выполнялись. • Условие «ИЛИ» требует, чтобы выполнялось хотя бы одно из указанных условий.

Условия

Указываются параметры (выражения) в составе условия. Каждый параметр добавляется в отдельном поле по кнопке +Добавить и удаляется по кнопке с минусом.

В разделе **Условия**:

1. Название параметра.
2. Логический оператор:
 - = – равно;
 - != – не равно;
 - < – меньше;
 - <= – меньше или равно;
 - > – больше;
 - >= – больше или равно;
 - * – содержит;
 - ** – совпадает.
3. Значение параметра, которое должно удовлетворять условию.

Пример:

1. Условие: {сумма_запроса} <= 100000 — Ветка «рассрочка до 100000».
2. Условие: {сумма_запроса} > 100000 — Ветка «кредит более 100000».
3. Ветка «по умолчанию» — Обработка других сценариев или ошибок.

□ Избегайте пересечения условий. Это предотвращает неоднозначность и неправильную обработку запросов.

Пример

1. Условие:

- Параметр: {сумма_запроса}.
- Логический оператор: > (больше).
- Значение: 50000.
- Ветка: «рассрочка более 50000».

2. Условие:

- Параметр: {сумма_запроса}.
- Логический оператор: > (больше).
- Значение: 30000.
- Ветка: «рассрочка более 30000, но не более 50000».

3. Условие:

- Параметр: {сумма_запроса}.
- Логический оператор: <= (меньше или равно).
- Значение: 70000.
- Ветка: «рассрочка не более 70000».

Ошибка:

Если значение {сумма_запроса} равно 40000, то все три условия выполняются, так как $40000 > 30000$, $40000 > 50000$ и $40000 \leq 70000$. В результате бот может выбрать все три ветки, что может привести к нежелательному поведению.

Исправление ошибки:

1. Используйте «и» вместо «или» в условиях:

- Измените второе условие на {сумма_запроса} > 30000 и {сумма_запроса} <= 50000. Теперь оно будет выполняться только для значений {сумма_запроса} в диапазоне от 30000 до 50000.

2. Исключите дублирующиеся диапазоны:

- Убедитесь, что диапазоны значений в условиях не дублируются. Например, если первая ветка обрабатывает значения более 50000, то вторая ветка должна начинаться с 50001, чтобы избежать пересечения.
-

Добавляйте ветку «по умолчанию» для ситуаций, когда ни одно из условий не выполнено.

<i>Переменные</i>	
Каждая переменная добавляется в отдельном поле по кнопке +Добавить и удаляется по кнопке с минусом.	
Название / Поле для ответа	<p>Названия переменных и значения, которые необходимо присвоить переменным.</p> <p>В названиях переменных разрешено использовать: кириллические буквы, латинские буквы, цифры, точку (.), нижнее подчеркивание (_), дефис (-), квадратные скобки ([]). Пробел использовать не рекомендуется.</p>

6.9.11. Выражение

Блок **Выражение** используется для описания логики поведения робота с помощью VPL-выражений — человекочитаемый код, предназначенный для решения сложных задач.

Из блока можно провести стрелки только в два блока.

Свойства блока

Чтобы перейти к свойствам, нажмите на блок. Справа откроется окно со свойствами.

Свойство	Описание
Подпись	Название блока, которое отображается в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства при создании сценария.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.
Количество циклов	Максимальное количество раз, которое робот может пройти по циклу. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.

ВРЛ-выражение	Чтобы ввести данные, нажмите кнопку Открыть редактор и воспользуйтесь формой Ввод ВРЛ-выражения, где в поле Содержимое ввода необходимо указать ВРЛ-выражение.
---------------	--

6.9.12. Пауза

Блок **Пауза** переводит систему в режим ожидания ответа – то есть, на паузу. Чтобы бот продолжил общение, он должен распознать одно из заданных слов для перебивания.

В отличие от блока **Вопрос**, блок **Пауза** позволяет ожидать ответ клиента продолжительное время

Из блока стрелка проводится только в один блок.

Свойства блока

Свойство	Описание
Основное	
ID	Уникальный ID блока.
Подпись	Название блока, которое отображается в блоке в редакторе сценария, но не используется в диалоге с клиентом. Служит для удобства работы со сценарием.
Маркер	Условное обозначение блока, произвольный текст. Может использоваться при аналитике в отчетах как маркер прохождения блока – с его помощью можно проследить, какие блоки прошел робот.

Сообщение	Текст, который будет воспроизведен роботом после перехода в блок. Отображается в блоке в редакторе сценария, если не задана Подпись, а также высвечивается как всплывающая подсказка при наведении на блок.
Задержка	Период времени, по истечении которого будет отправлено сообщение после перехода в блок. С помощью задержки перед отправкой можно симитировать набор сообщения – как будто с клиентом общается не бот, а реальный человек. Задается в секундах.
Количество циклов	Максимальное количество раз, которое бот может пройти по циклу, после бот переведет на оператора. Циклом считается повторное попадание пользователем в конкретный блок. Максимальное количество циклов – 100.
Немедленная отправка	Позволяет отправить сообщения до полного формирования очереди в связке с блоком запроса к серверу.

Активное слушание

Это функция голосовых ботов, которая делает диалог более естественным. Она вставляет синтезированные или записанные фразы в ответ на речь пользователя, имитируя поведение живого собеседника. Подробнее об активном слушании читайте в статье [Активное слушание](#).

Чтобы настройки активного слушания использовались для указанного блока, нажмите + Добавить активное слушание. Если же вы не хотите применять эту функцию к данному блоку, нажмите переключатель. При удалении активного слушания для блока будут использоваться настройки стартового блока.

Таким образом, разница между отключением и удалением заключается в том, что при отключении активное слушание не используется в конкретном блоке, но его настройки сохраняются, а при удалении настройки блока сбрасываются на настройки стартового блока.

Мин. интервал озвучки (мс)	Начало интервала от начала речи пользователя, в течение которого можно запускать фразы озвучки.
Макс. интервал озвучки (мс)	Конец интервала, в течение которого можно запускать фразы озвучки.
Фразы озвучки	Список фраз, которые будут озвучены пользователю в случайное время в указанном выше промежутке. Фразы произносятся в случайном порядке. Фразы добавляются по кнопке Добавить и удаляются по кнопке  .
<i>Максимальное время ожидания</i>	
<input type="checkbox"/> Используется только в телефонии.	
Ожидание до повтора	Максимальное время ожидания в секундах, через которое будет произнесен текст из поля Текст повтора.
Ожидание ответа собеседника	Максимальное время ожидания ответа клиента в секундах, по истечении которого произойдет перевод на оператора.
Текст повтора	Варианты повтора, которые будут озвучены роботом по истечении времени ожидания, заданного в поле Ожидание до повтора. Каждая фраза должна быть вписана в отдельном поле. Поля для фраз добавляются по кнопке Добавить и удаляются по кнопке  . Если вариантов больше одного, фразы будут произноситься в случайном порядке.

Фиксация результата

Используется для сохранения данных диалога в отчет, в котором будут заполнены соответствующие полям столбцы. Чтобы фиксировать результат, активируйте переключатель.

Дата	Дата фиксации для выгрузки в отчет.
Оценка	Оценка действия для выгрузки в отчет.
NPS	Произвольная информация (переменная, ответ пользователя и т.д.), которая будет использоваться для анализа диалогов. Записывается в отчеты Сессии и Сессии с маркерами в поле NPS.
Подтверждение	Краткий смысл происходящего в данном блоке. Например, «Уточняем у клиента цвет авто». Записывается в отчеты Сессии и Сессии с маркерами в поле Результаты диалога с ботом.

Перебивания

Используется для того, чтобы клиент мог прерывать высказывание бота в диалогах, используя слова для перебивания. При произнесении этих слов во время высказывания бота, его речь завершается, и происходит переход к следующему этапу сценария.

Уровень перебивания	Настройка перебивания бота в диалогах. Читайте об уровнях перебивания в статье Перебивания .
Игнорировать вышестоящие перебивания	Эта функция позволяет игнорировать установленные слова для перебивания на вышестоящих уровнях, делая актуальными лишь слова, указанные в этом блоке и ниже.

+ Добавить список перебиваний	Кнопка, с помощью которой добавляются слова для перебивания.
-------------------------------	--

6.10. Системные переменные в сценарии

На платформе существуют системные переменные, которые можно использовать в любом сценарии. Системные переменные не нужно объявлять и записывать в них данные — вы можете сразу использовать их в своих сценариях.

При указании переменных использовать фигурные скобки необходимо только при указании переменных в сообщениях, которые воспроизводятся ботом. В остальных случаях фигурные скобки не используются.

6.10.1. Список системных переменных

Название	Описание
{now}	В звонке переменная возвращает дату и время в соответствии с часовым поясом номера клиента, а в чате — по всемирному времени UTC.
{dialogId}	Возвращает строку с идентификатором текущего диалога.
{dialog}	Возвращает строку с полным текстом диалога.
{previousBotId}	Возвращает ID сценария, из которого система перенаправила бота в другой сценарий во время диалога с пользователем.
{today}	Возвращает дату без указания времени (например, 2023- 11-16). Текущая дата определяется во время диалога с клиентом.
{time}	Возвращает время без указания даты (например, 15:53:12). Текущее время определяется во время диалога с клиентом.
{phone}	Возвращает номер телефона клиента, который указан в задании на обзвон для выполнения вызова.
{recordPath}	Возвращает ссылку на аудиозапись текущего диалога в телефонии.
{percept}	Возвращает строку с последним ответом клиента на вопрос бота.
{tryNumber}	Возвращает количество попыток дозвона в рамках задания на обзвон.

{company.now}	Возвращает дату и время с учетом часового пояса, в котором расположена компания (например, 2023-11-17 13:06:58). Текущие дата и время определяются во время диалога с клиентом.
{company.today}	Возвращает дату без указания времени (например, 2023- 11-17). При этом учитывается часовой пояс, в котором расположена компания. Текущая дата определяется во время диалога с клиентом.
{company.time}	Возвращает время без указания даты (например, 13:06:58). При этом учитывается часовой пояс, в котором расположена компания. Текущее время определяется во время диалога с клиентом.
{companyId}	Возвращает строку с идентификатором компании, которой принадлежит бот.
{botId}	Возвращает строку с идентификатором бота.
{referrer}	Возвращает строку с адресом главной страницы сайта, на которой размещен виджет (позволяет оператору понять, с какого сайта пишет пользователь).
{utcNow}	Возвращает дату и время в часовом поясе по UTC.
{utcTime}	Возвращает время в часовом поясе по UTC.
{utcToday}	Возвращает дату в часовом поясе по UTC.
{messengerUserId}	Возвращает идентификатор учетной записи пользователя в мессенджере.
{communicationType}	Строка, тип коммуникации с клиентом. Допустимые значения: TEXT, VOICE.
{channelType}	Строка, тип канала общения. Допустимые значения: WEB, MOBILE, MESSENGER.
{messenger}	Строка, тип мессенджера. Допустимые значения: WHATSAPP, VIBER, TELEGRAM, VKONTAKTE, FACEBOOK1, SKYPE, SLACK, YANDEX, ALICE, THREADS.
{clientLastAnswerTime}	Возвращает время последнего ответа пользователя на вопрос бота. Значение отображается в виде временной метки Unix-времени ² .
{botLastAnswerTime}	Возвращает время последнего ответа бота в секундах. Значение отображается в виде временной метки Unix-времени ² .
{clientId}	Возвращает строку с внутренним идентификатором клиента в системе TWIN (uuid).
{clientPhone}	Возвращает строку с номером телефона клиента.

{clientName}	Возвращает строку с именем клиента.
{clientNickname}	Возвращает имя клиента (никнейм) в мессенджере, в котором клиент общался с ботом.
{clientMetadata}	Возвращает массив, содержащий любые данные о клиенте (задается в следующем формате: {{firstName}}: «Иван», {{middleName}}: «Иванович», {{lastName}}: «Иванов»).

1 Деятельность организации Meta Platforms Inc. и ее продуктов Instagram и Facebook запрещена в Российской Федерации.

2 Временная метка Unix-времени (например, 1658125300) отображает количество секунд, которое прошло с 1 января 1970 года (00:00:00 UTC).

6.11. Средства форматирования даты и времени

При помощи программных средств можно отображать дату и время в желаемом формате. Например, клиент может написать в диалоге следующую фразу: «третьего октября в восемь». Ответ клиента можно сохранить в переменную, а в дальнейшем (при необходимости) отобразить в диалоге в нужном виде, например: 2022-10-03 в 08:00, 08:00 03-10, 22-10-03, 03.10.22, в 08:00, 08:00 AM, Monday и др.

Программные средства форматирования даты и времени можно использовать в сценариях в виджетах, социальных сетях и мессенджерах. В телефонии форматирование даты и времени не поддерживается.

По умолчанию дата и время отображаются в сценариях в следующих форматах:

- Дата: ГГГГ-ММ-ДД
- Время: ЧЧ:ММ:СС
- Дата и время: ГГГГ-ММ-ДД ЧЧ:ММ:СС

Для форматирования дат и времени служит следующий шаблон: {<имя переменной>|формат: код или коды форматирования}.

6.11.1. Список кодов форматирования

В качестве примера ниже используются следующие дата и время: «3 октября 2022, 8:15».

Код форматирования	Описание / обозначение	Пример использования	Результат
a	«am» или «pm»	{имя переменной формат:a}	am

A	«AM» или «PM»	{имя переменной формат:A}	AM
d	День месяца (01-31)	{имя переменной формат:d}	03
D	Сокращенное название дня недели (первые три буквы)	{имя переменной формат:D}	Mon
F	Полное название месяца	{имя переменной формат:F}	October
g	Часы (12-часовой формат без ведущих нулей)	{имя переменной формат:g}	8

Код форматирования	Описание / обозначение	Пример использования	Результат
G	Часы (24-часовой формат без ведущих нулей)	{имя переменной формат:G}	8
h	Часы (12-часовой формат)	{имя переменной формат:h}	08
H	Часы (24-часовой формат)	{имя переменной формат:H}	08
i	Минуты (00-59)	{имя переменной формат:i}	15
j	День месяца без ведущих нулей (1-31)	{имя переменной формат:j}	3
l	Полное название дня недели	{имя переменной формат:l}	Monday
L	Признак високосного года (0	{имя переменной формат:L}	0

	– обычный год или 1 – високосный год)		
m	Номер месяца (01-12)	{имя переменной формат:m}	10
M	Сокращенное название месяца (три буквы)	{имя переменной формат:M}	Oct
n	Месяц (1-12)	{имя переменной формат:n}	10
s	Секунды (00-59)	{имя переменной формат:s}	00
t	Количество дней в данном месяце (28-31)	{имя переменной формат:t}	31
w	Номер дня недели (0 – воскресенье, 6 – суббота)	{имя переменной формат:w}	1
y	Год (последние два разряда)	{имя переменной формат:y}	22
Y	Год (четыре разряда)	{имя переменной формат:Y}	2022
z	Номер дня в году (0-365)	{имя переменной формат:z}	275

Вы можете использовать одновременно несколько кодов форматирования из списка выше. В качестве разделителя можно использовать различные символы: точка, запятая, точка с запятой, двоеточие, плюс, минус и прочие, или не использовать разделитель вообще. В строке форматирования также можно использовать различные слова и фразы.

Например:

Строка форматирования	Результат
{имя переменной формат:l d.m.y}	Monday 03.10.22

Строка форматирования	Результат
-----------------------	-----------

{имя переменной формат: текущий день: l, текущая дата: d.m.y}	Текущий день: Monday, текущая дата: 03.10.22
{имя переменной формат:h:i:s}	08:15:00
{имя переменной формат:g часов i минут s секунд}	8 часов 15 минут 00 секунд

6.11.2. Склонение значений пользовательских переменных по падежам

- {переменная|падеж:имя_падежа} – формат записи имен пользовательских переменных, который позволяет склонять их значения по падежам.
- Пример использования:
- {userName|падеж:родительный} – где userName, например: Иван Иванович.
- Например, требуется задать вопрос: «Могу я услышать Ивана Ивановича?». Чтобы бот использовал значение переменной в родительном падеже, в сообщении, которое будет озвучено ботом в блоке Вопрос, нужно указать следующее: «Могу я услышать {userName|падеж:родительный}?»

Поддерживаются все падежи:

- Именительный (кто? что?) – пример: «это кто?» – «Иван Иванович»; Родительный (кого? чего?) – пример: «нет кого?» – «Ивана Ивановича»; Дательный (кому? чему?) – пример: «кому вы рады?» – «Ивану Ивановичу»;
- Винительный (кого? что?) – пример: «кого/что вы видите?» – «Ивана Ивановича»; Творительный (кем? чем?) – пример: «кем восхищаетесь?» – «Иваном Ивановичем»; Предложный (о ком? о чем?) – пример: «о ком думаете?» – об «Иване Ивановиче».

6.12. Таймеры и принципы в распознавании речи

Параметры таймеров определяют решение об окончании фразы (сеанса распознавания). Кроме того, система распознавания (ASR) может самостоятельно вернуть признак, что клиент закончил говорить. Она это делает при помощи обученной нейронной модели и реагирует на интонации, языковые конструкции и прочие закономерности, которые нашла в файлах для обучения.

Если ASR вернула признак конца фразы, остальные параметры таймеров будут игнорироваться. Даже если клиент закончил одну фразу и начал говорить вторую практически мгновенно и успел по всем параметрам времени продолжить свою речь, система уже не будет его слушать. Она получила признак окончания фразы и завершила сеанс распознавания, поэтому игнорирует все указанные ниже параметры.

Длительность сеанса распознавания речи определяется следующими параметрами:

- **sint (speech incomplete timeout)** — промежуток тишины между двумя словами.
- **nit (no input timeout)** — время ожидания ответа.
- **t (recognition timeout)** — период распознавания ответа.

В системе предусмотрено 10 уровней длительности сеанса распознавания:

Уровень	sint (мс)	nit (мс)	t (мс)
Односложный ответ	100	4000	7000
Очень очень короткая	300	2000	5000
Очень короткая	300	3000	5000
Короткая	400	3000	5000
Нормальная	960	3000	7000
Нормальная (5 сек.)	960	5000	7000
Нормальная (180 сек.)	1200	3000	180000
Длинная	1200	4000	10000
Очень длинная	3000	4000	15000
Очень длинная (180 сек.)	3000	4000	180000

Определение времени ответа бота

Время, через которое ответит бот, не равно длительности сеанса распознавания. Время до ответа бота высчитывается по следующей формуле: получение ответа от ASR + время на принятие решения. Прочие звуки после основной речи клиента (до наступления тишины) и пауза в аудиофайле бота перед началом воспроизведения, могут увеличить время ответа бота.

Пример расчета времени ответа бота

Длительность сеанса распознавания — **Короткая**. После наступления тишины и завершения сеанса распознавания потребовалось 0.1 сек. на получение полного ответа от ASR. Прежде чем продолжить, боту потребовалось еще 0.3 сек. на принятие решения. Аудиофайл ответа бота содержал в себе паузу перед началом речи продолжительностью 0.1 сек.

Таким образом, общая длительность паузы составляет: $0.3 + 0.1 + 0.3 + 0.1 = 0.8$ сек.

6.13. Активное слушание

Активное слушание — это функционал для голосовых ботов, который позволяет им реагировать на речь пользователя, делая диалог более естественным и

вовлеченным. Бот использует синтезированные фразы или заранее записанные фразы озвучки, чтобы реагировать на речь пользователя, имитируя поведение живого собеседника.

Цель активного слушания — сделать общение с голосовым ботом более человечным и естественным. Когда бот вставляет фразы подтверждения, пользователь чувствует, что его слушают и понимают. Это повышает удовлетворенность и доверие к системе, а также способствует более плавному и непрерывному диалогу.

Принципы работы

1. Активное слушание включается, когда бот распознает начало речи пользователя — это может быть вызвано триггерным словом, командой или обращением. Система продолжает активное слушание в ходе диалога, отслеживая паузы и изменения в речи, чтобы поддерживать плавное взаимодействие.
2. Во время разговора бот вставляет короткие фразы, такие как «угу», «да, я понимаю», «продолжайте». Эти фразы могут быть синтезированы или воспроизведены заранее записанным голосом, в зависимости от настроек. Это помогает пользователю понять, что бот продолжает слушать и следить за диалогом, создавая ощущение живого общения. Механизм работы:
 - Первую фразу бот произносит не по таймеру, а при наличии нескольких одинаковых промежуточных результатов распознавания. Например, если бот видит, что пользователь говорит длинную фразу или несколько предложений подряд, он может сказать «угу», показывая, что понимает и слушает. Это может случиться как раньше, так и позже таймеров.
 - Бот никогда не говорит в тишину — реакция возникает только на речь.
 - После первой фразы бота система начинает следовать определенным интервалам для следующих реакций, описанным ниже.
3. После того как бот выдал первое подтверждение, в работу вступают таймеры:
 - Минимальный интервал — это время, через которое бот может снова выдать фразу подтверждения. Например, если установлен интервал в 500 мс, бот не будет говорить «угу» или другие фразы до истечения этого времени.
 - Гибкость таймеров — бот может озвучить подтверждение позже минимального интервала, если пауза в речи пользователя затянулась, но не раньше установленного времени. Например, пользователь говорит длинную фразу, бот говорит «угу». Далее бот ждет, пока пройдет минимальный интервал (например, 500 мс), прежде чем снова может подтвердить, что слушает, если речь продолжается.
4. В зависимости от контекста диалога и целей сценария, можно изменить частоту и содержание фраз для создания более естественного взаимодействия. Например, в формальном сценарии можно использовать менее частые и более нейтральные фразы, а в неформальной беседе — более оживленные и частые подтверждения.

5. Бот не использует фразы подтверждения в случае полной тишины. Это предотвращает ощущение неестественного поведения или того, что бот «завис». Фразы подтверждения всегда зависят от распознанной речи или ее промежуточных результатов.

Использование активного слушания

Функционал доступен в следующих блоках сценария:

1. **Настройки** — активное слушание будет работать для всех блоков Вопрос и Пауза в сценарии. Подробнее о блоке и настройках активного слушания читайте в статье [Настройки](#).
2. **Вопрос** — активное слушание будет работать только в конкретном блоке Вопрос. Подробнее о блоке и настройках активного слушания читайте в статье [Вопрос](#).
3. **Пауза** — активное слушание будет работать только в конкретном блоке Пауза. Подробнее о блоке и настройках активного слушания читайте в статье [Пауза](#).

□ Для более естественного и качественного диалога голосового бота с пользователем рекомендуется использовать предзаписанные фразы, а не синтез речи. Это связано с тем, что:

- синтезированная речь может быть громкой;
- синтез не сможет воспроизвести междометия и другие короткие эмоциональные реакции, такие как «угу», которые важны для поддержания беседы.

В результате синтезированные ответы могут звучать неестественно, поэтому предпочтительнее использовать заранее записанные фразы.

Поддержка перебиваний

Активное слушание поддерживает работу с включенными [перебиваниями](#), что делает бота еще более интерактивным и способным реагировать на изменения в диалоге в реальном времени.

6.14. Регулярные выражения

Регулярные выражения — это инструмент для поиска, замены и проверки текста.

Они представляют собой особый язык, который позволяет задавать шаблоны и искать или проверять текст, соответствующий этим шаблонам.

Регулярные выражения состоят из специального набора символов, которые обозначают определенные шаблоны. Вот основные элементы регулярных выражений:

- **Символы** — буквы, цифры и другие символы.
- **Мета-символы** — специальные символы, которые имеют особое значение. Например, `.` для любого символа, `\d` для цифры.

- **Квантификаторы** — указывают, сколько раз элемент может появляться. Например, `*` для нуля или более раз, `+` для одного или более раз.
- **Группы и диапазоны** — позволяют объединять символы и задавать диапазоны. Например, `[A-Za-z]` для любой буквы.

Спецсимволы в регулярных выражениях

Список некоторых часто используемых специальных символов в регулярных выражениях:

- `\d` — соответствует любой цифре от 0 до 9.
- `\w` — соответствует любому символу латиницы, цифре или символу подчеркивания (в ASCII).
- `\s` — соответствует любому пробельному символу.
- `\b` — обозначает границу слова.
- `^` — соответствует началу строки.
- `$` — соответствует концу строки.
- `.` — соответствует любому символу, кроме перевода строки.
- `*` — повторяет предшествующий символ 0 или более раз.
- `+` — повторяет предшествующий символ 1 или более раз.
- `?` — предшествующий символ встречается 0 или 1 раз.
- `{n}` — точное количество повторений предшествующего символа, где `n` — число.

Экранирование символов — способ, при котором специальные символы в регулярных выражениях (например, `.`, `*`, `+`) перестают работать как команды и становятся обычными символами. Чтобы программа поняла их как обычные знаки, а не как инструкции, перед ними ставят обратный слеш (`\`).

Пример экранирования:

Если нужно найти точку в тексте, вместо того чтобы использовать просто `.`, следует экранировать символ точкой с обратным слешем `\.`. Например, регулярное выражение `\.` будет искать именно точку, а не любой символ.

Валидация данных через регулярные выражения

Проверка валидности номера телефона

Если номер телефона должен вводиться с +7, используйте регулярное выражение, которое проверяет наличие этого префикса и далее последовательность цифр.

Пример подходящего регулярного выражения и кода для его проверки:

```
^+7\d{10}$
```

Где:

- `^` — начало строки.
- `+7` — указывает, что номер должен начинаться с +7. Символ `+` является специальным и означает один или более предыдущего символа. Чтобы использовать его в буквальном значении, как часть телефонного кода +7,

необходимо его экранировать: `\+`. В выражении `^+7\d{10}$` символ `+` будет воспринят как специальный, поэтому работа будет происходить некорректно.

- `\d{10}` — означает, что после `+7` должно идти ровно 10 цифр.
- `$` — конец строки.

Для более сложных требований к формату номера телефона, например, учитывать код страны или различные разделители, шаблон регулярного выражения можно усложнить.

Пример для проверки номера телефона с кодом страны:

```
^+?\d{1,3}[-.\s]?(\d{1,4}?)?\d{1,4}\d{1,9}$
```

Где:

- `^+?` — начало номера с символа «+». Символ `?` означает, что предшествующий символ может встречаться либо 0, либо 1 раз. Экранированный символ `+` и `?` говорят о том, что номер может как начинаться с символа `+`, так и обходиться без него. Например, выражение может захватить номера как в формате `+71234567890`, так и в формате `71234567890`.
- `\d{1,3}` — код страны из 1-3 цифр.
- (Опционально) `[-.\s]?` — разделитель: тире, точка, пробел.
- (Опционально) `(\d{1,4}?)?` — код региона в скобках из 1-4 цифр.
- `\d{1,4}` — 1-4 цифры.
- `\d{1,9}` — основная часть номера из 1-9 цифр.

Проверка валидности имени

Для проверки имени используйте регулярное выражение, которое позволяет только буквы и некоторые специальные символы. Например, дефис или апостроф.

Пример простого регулярного выражения для проверки имени:

```
^[A-Za-zА-Яа-яёЁ]+([-'][A-Za-zА-Яа-яёЁ]+)*$
```

Где:

- `^` — начало строки.
- `[A-Za-zА-Яа-яёЁ]+` — указывает, что имя должно начинаться с одной или более букв (латиница или кириллица, включая букву «ё»).
- `([-'][A-Za-zА-Яа-яёЁ]+)*` — указывает, что имя может содержать дефис или апостроф, за которым следует одна или более букв, и это может повторяться несколько раз.
- `$` — конец строки.

Проверка общего формата текста

Проверка валидности текста может включать проверку правильности формата, содержания и длины текста. Ниже приведены несколько распространенных сценариев проверки валидности текста и примеры регулярных выражений для каждого из них.

Чтобы убедиться, что текст соответствует определенному формату, например, содержит только буквы, пробелы и знаки препинания, используйте:

```
^[A-Za-zА-Яа-яёЁ\s]+
```

Где:

- `^` — начало строки.
- `[A-Za-zА-Яа-яёЁ\s]+` — один или более символов, включающих буквы и пробелы.
- `$` — конец строки.

Проверка общего формата текста с пунктуацией

Регулярное выражение для проверки, что строка начинается с заглавной буквы и заканчивается знаком пунктуации, может выглядеть так:

```
^[A-ZА-Я].*[!/?]
```

Где:

- `^[A-ZА-Я]` — строка должна начинаться с заглавной буквы (латиница или кириллица).
- `*` — любая последовательность символов (0 или более).
- `[!/?]` — строка должна заканчиваться точкой, вопросительным или восклицательным знаком.

Проверка длины текста

Для проверки текста длиной от 1 до 100 символов используйте следующий шаблон:

```
^{1,100}
```

Где:

- `^` — начало строки.
- `{1,100}` — количество от 1 до 100 раз. Укажите свой диапазон.
- `$` — конец строки.

Проверка наличия определенных слов или фраз

Чтобы убедиться, что текст содержит определенные слова или фразы, используйте:

```
\bexample\b"
```

Где:

- `\b` — граница слова.
- `example` — слово, которое должно присутствовать в тексте. Укажите свое слово.
- `\b` — граница слова.

□ Используйте только латинские буквы.

Валидация URL

Для проверки правильности формата URL используйте:

```
^(https?:\W)?([\w-]+\.)+[\w-]+(\.[\w-\.]*)*V?$
```

Где:

- `^` — начало строки.
- (Опционально) `(https?:\W)?` — http или https, за которым следует «: //».
- `([\w-]+\.)+` — одно или более слов / дефисов, за которыми следует точка.
- `[\w-]+` — одно или более слов / дефисов.
- `(\.[\w-\.]*)*` — ноль или более раз символ «/», слова, дефисы или точки.
- (Опционально) `V?` — символ «/» в конце строки.
- `$` — конец строки.

6.15. Управление заданиями на обзвон

Работа с заданиями на обзвон (настройка телефонии) происходит в разделе **Голосовые боты** на вкладке **Задания**.

6.15.1. Работа со списком заданий на обзвон

Для каждого задания отображаются:

- **Название** — наименование, которое было присвоено заданию при его [создании](#). По названию вы можете быстро идентифицировать нужное задание в списке.
- **CPS** — частота, с которой проводится обзвон; количество звонков клиентам в секунду. Задается при [создании задания](#) в параметре Интенсивность дозвона.
- **Кандидаты** — количество потенциальных клиентов, прикрепленных к заданию на обзвон, которым и звонит бот. Подробнее в статье [Работа со звонками в задании на обзвон](#).
- **Номера** — количество номеров клиентов, по которым проводится обзвон. Может быть равным или превышать число кандидатов, если для них указано несколько номеров. Подробно о добавлении номеров см. [Работа со звонками в задании на обзвон](#).
- **Режим запуска** — режим запуска, выбранный для задания на обзвон. Возможны следующие варианты:
 - **Вручную** — пользователь запускает обзвон вручную в удобное для него время. Вызовы запускаются по кнопке запуска справа в строке задания на обзвон.
 - **По расписанию** — система запускает обзвон в заданное вами время автоматически. Под статусом указываются дата и время следующего обзвона.

- **Статус** — статус выполнения задания на обзвон: Создано, Готово к запуску, Выполняется, Превышен лимит, Пауза, Остановлено, Выполнено, Закончились деньги, Ошибка.
- **Обновлен** — дата и время, когда было создано или внесено последнее изменение в задание.

Название	CPB	Кандидаты	Номера	Прогресс	Статус	Режим запуска	Обновлен
Call_1	1	2	2	100%	Выполнено	Вручную	06.06.24, 15:04
Call_2	1	0	0	0%	Создано	Вручную	06.06.24, 14:39
Call_3	1	2	2	100%	Выполнено	Вручную	12.12.23, 04:00
Call_4	1	0	0	0%	Остановлено	Вручную	01.12.23, 05:06
Call_5	1	1	1	100%	Остановлено	Вручную	01.12.23, 05:06
Call_6	1	3	3	100%	Выполнено	Вручную	17.11.23, 21:21
Call_7	1	4	4	100%	Выполнено	Вручную	17.11.23, 21:01
Call_8	1	4	4	75%	Пауза	Вручную	17.11.23, 17:48
Call_9	1	2	2	100%	Выполнено	Вручную	17.11.23, 17:37

Вы можете быстро найти нужное задание, воспользовавшись следующими функциями:

1. Поиск задания в списке.
2. Сортировка списка заданий.
3. Фильтрация по статусу задания на обзвон.

6.15.2. Поиск задания

Нужное задание можно быстро найти при помощи поля поиска, которое расположено над списком заданий. Вы можете ввести полное название задания или его часть. Результаты поиска отображаются автоматически. Для сброса поиска необходимо вручную удалить значение из поля поиска.

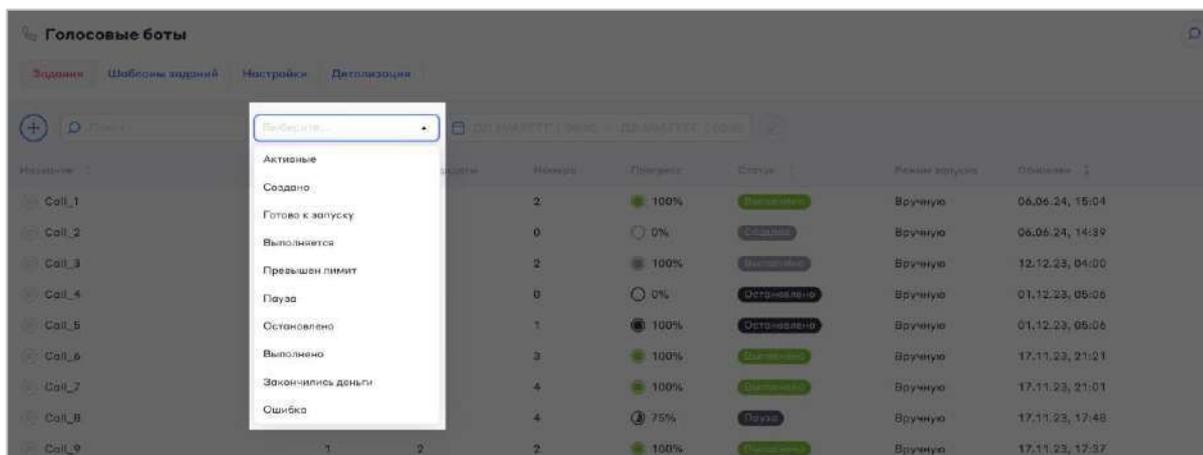
6.15.3. Сортировка списка заданий

Список заданий можно сортировать по названию задания: в прямом или обратном алфавитном порядке, а также по дате обновления задания: от более поздней к более ранней и наоборот.

По умолчанию задания отсортированы по дате обновления от более поздней к более ранней.

6.15.4. Фильтрация по статусу задания на обзвон

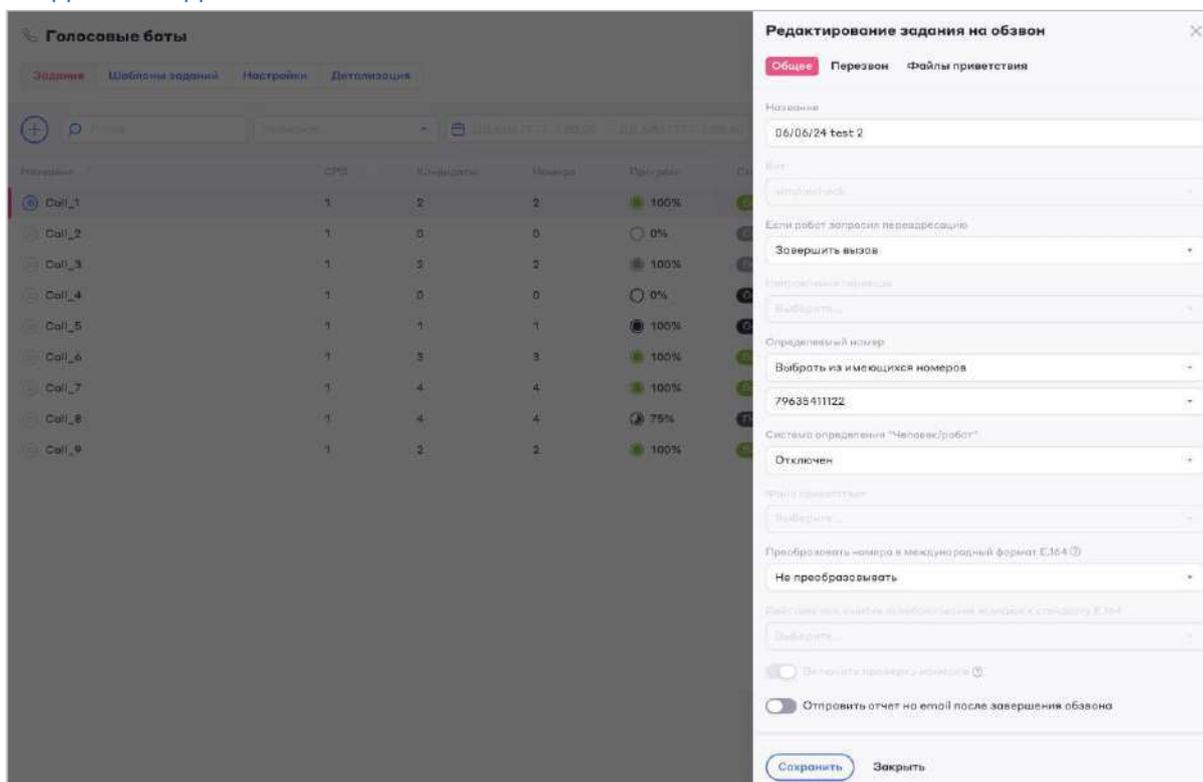
Вы можете вывести только задания на обзвон с конкретным статусом, выбрав этот статус в выпадающем списке возле поля поиска. Чтобы сбросить статус, нажмите на крестик в поле фильтрации.



6.15.5. Редактирование задания на обзвон

Чтобы перейти к редактированию задания, наведите курсор на строку задания и нажмите на появившуюся справа кнопку с многоточием и далее – на кнопку **Редактировать**.

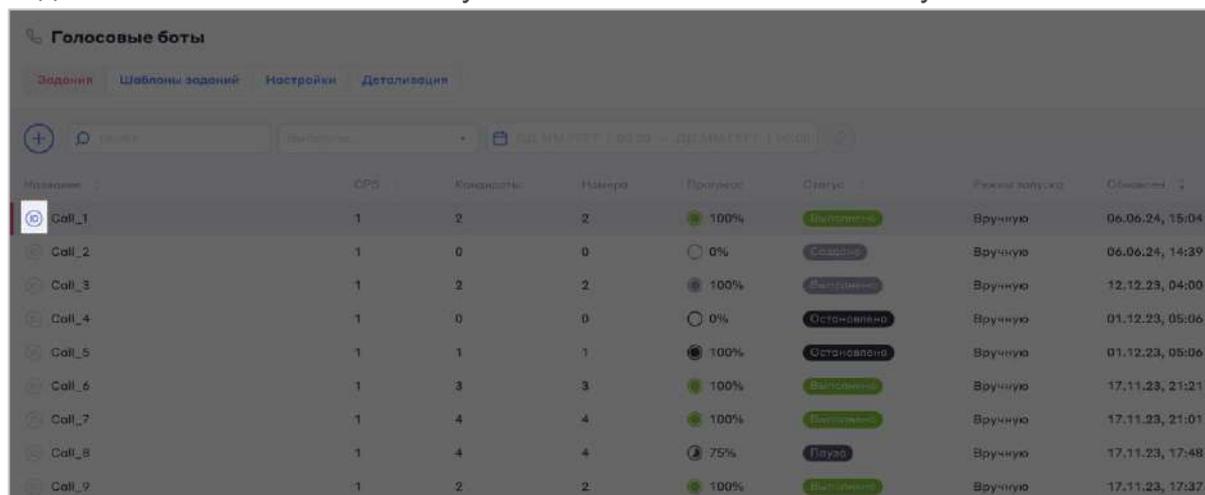
Действия по редактированию задания на обзвон полностью аналогичны действиям по [созданию задания](#).



6.15.6. Копирование идентификатора задания на обзвон

Копирование идентификатора может быть полезно, например, для передачи его в техническую поддержку.

Чтобы скопировать идентификатор задания на обзвон, наведите курсор на строку задания и нажмите на появившуюся в колонке **Название** иконку **ID**.



Название	СРБ	Кандидаты	Номера	Прогресс	Статус	Режим запуска	Обновлен
Call_1	1	2	2	100%	Выполнено	Вручную	06.06.24, 15:04
Call_2	1	0	0	0%	Создан	Вручную	06.06.24, 14:39
Call_3	1	2	2	100%	Выполнено	Вручную	12.12.23, 04:00
Call_4	1	0	0	0%	Остановлено	Вручную	01.12.23, 05:06
Call_5	1	1	1	100%	Остановлено	Вручную	01.12.23, 05:06
Call_6	1	3	3	100%	Выполнено	Вручную	17.11.23, 21:21
Call_7	1	4	4	100%	Выполнено	Вручную	17.11.23, 21:01
Call_8	1	4	4	75%	Пауза	Вручную	17.11.23, 17:48
Call_9	1	2	2	100%	Выполнено	Вручную	17.11.23, 17:37

После этого идентификатор будет скопирован в буфер обмена на вашем компьютере.

6.15.7. Запуск задания на обзвон

Чтобы запустить вызовы вручную, используйте кнопку запуска справа в строке задания на обзвон. Необходимо запускать те вызовы, для которых установлен **Режим запуска – Вручную**, а также возобновлять перезапущенные неуспешные вызовы в заданиях. Запустить задание на обзвон можно, только когда были добавлены кандидаты на обзвон. При отсутствии кандидатов кнопка будет неактивна.

6.15.8. Звонки задания на обзвон

Переход к звонкам задания на обзвон осуществляется по нажатию на любое место в строке задания на обзвон (кроме прочих кнопок).

6.15.8. Удаление задания на обзвон

Чтобы перейти к удалению задания на обзвон, наведите курсор на строку задания на обзвон и нажмите на появившуюся справа кнопку с многоточием и далее – на кнопку **Удалить**.

В появившемся всплывающем окне подтвердите удаление по кнопке **Удалить**.

Голосовой бот будет удален с его настройками и звонками без возможности восстановления!

6.16. Создание задания на обзвон

1. Перейдите в раздел **Голосовые боты** личного кабинета.
2. На вкладке **Задания** нажмите **Создать задание на обзвон** или  в правом верхнем углу. Откроется окно **Создание задания на обзвон**.
3. На вкладке **Общее** выполните следующие настройки:

- **Название:** введите наименование задания на обзвон. Выбирайте простые и понятные названия. В будущем это облегчит вам поиск нужного задания на обзвон.
- **Бот:** из выпадающего списка выберите **сценарий**, который будет использоваться в диалогах с клиентами.
- **Если бот запросил переадресацию:** из выпадающего списка выберите одно из следующих значений:
 - **Завершить вызов** — вызов будет завершен.
 - **Ничего не делать** — робот прекратит свою работу, вызов будет завершен.
 - **Передать вызов на канал** — будет выполнена переадресация на предустановленное направление перевода. При выборе данного значения, в поле Направление перевода выберите нужное направление перевода. Добавить новое направление перевода вы можете в разделе Голосовые боты на вкладке **Настройки** → **Переводы**.
- **Определяемый номер:** из выпадающего списка выберите одно из значений:
 - **Настройки по умолчанию** — не определено.
 - **Выбрать из имеющихся номеров** — определенный номер, зарегистрированный у оператора связи, который будет отображаться у клиента во время вызова. В дополнительном поле из выпадающего списка выберите нужный номер. Добавляются номера в разделе Голосовые боты на вкладке **Настройки** → **Управление номерами**.
 - **Выбрать преднастроенный пул номеров** — во время вызова у клиента будет отображаться один (любой) номер из пула CallerID номеров, зарегистрированных у оператора связи. В дополнительном поле из выпадающего списка выберите нужный пул. Создать новый пул вы можете в разделе Голосовые боты на вкладке **Настройки** → **Пулы CallerID**.
- Система определения «Человек/робот»: из выпадающего списка выберите нужное значение:
 - **Отключен** — система отключена. Робот продолжает диалог в любом случае.
 - **Фоновый режим определения** — определение происходит в фоновом режиме во время звонка. Если система определяет, что в разговоре участвует автоответчик, то вызов автоматически завершается.
 - **Определение с блокировкой** — голосовой робот не начнет разговор до тех пор, пока система не определит, что в вызове участвует человек. Для определения используется Файл приветствия. При выборе Определения с блокировкой, в поле Файл приветствия выберите нужный файл. Файл проигрывается сразу после звонка клиенту. Если на звонок отвечает человек, система подключает бота, в противном случае звонок завершается. Добавить новый файл приветствия вы можете во вкладке Файлы приветствия. Загружаемый файл должен иметь расширение mp3 и не должен превышать 10 Мб.
- **Преобразовать номера в международный формат E.164:** из выпадающего списка выберите нужное значение:

- **Не преобразовывать** — номера не будут преобразовываться в [международный формат E.164](#).
- **Для номеров из России** — номера из России будут преобразованы в международный формат E.164. При выборе данного значения, в поле Действие при ошибке преобразования номеров к стандарту E.164 выберите значение Не добавлять номера или Пометить добавленные номера. Подробнее о функции читайте в статье [Нормализация](#).
- (Опционально) Включите **Включить проверку номеров** — система проверяет корректность формата номера при добавлении кандидата. Проверка работает как для российских номеров, так и для иностранных. При необходимости отключите данный параметр.
- (Опционально) Включите **Отправить отчет на email после завершения обзвона** — после завершения обзвона система отправит отчет на email компании, указанный в личном кабинете. При необходимости отключите данный параметр.
- Режим запуска задания: из выпадающего списка выберите нужный режим:
 - **Вручную** — задание запускается вручную с помощью кнопки Продолжить обзвон.
 - **В указанное время** — задание на обзвон запускается автоматически. В дополнительном поле при помощи календаря выберите желаемую дату и время запуска задания на обзвон.
- Интенсивность дозвона: настройте скорость, с которой будут совершаться новые звонки. Например, если вы выбрали 3 попытки в 1 секунду, это значит, что будет совершаться 3 звонка каждую секунду.
- (Опционально) Настройте **Ограничитель обзвона по времени суток**. Передвигая ползунки, выберите желаемый временной диапазон, в течение которого нужно выполнить обзвон (время начала первого и последнего звонков). Система TWIN определяет часовые пояса кандидатов на обзвон и совершает звонки в указанный промежуток. При необходимости отключите данный параметр.
- Укажите **Длительность результативного вызова, более**. Введите минимальную длительность вызова в секундах, при которой звонок будет считаться результативным. Если продолжительность звонка будет меньше указанной, вызов будет считаться нерезультативным. Используется при формировании отчетов, а также может использоваться при настройке перезвонов.
- (Опционально) **Адрес вебхука**: введите URL-адрес [вебхука](#), на который вы хотите получать статистику по вызовам.
- (Опционально) **Комментарий к заданию**: введите информацию, которая может быть полезна.

4. (Опционально) Перейдите во вкладку **Перезвон**. Здесь вы можете задать правила, которые будут определять условия выполнения повторного вызова клиенту. Для этого:

- Включите **Использовать стратегию перезвона**.
- Стратегия обзвона кандидатов: из выпадающего списка выберите нужную стратегию:
 - **Последовательный** — для кандидата с несколькими номерами сначала осуществляются все попытки перезвонить на первый номер, затем производятся все попытки дозвона на следующий его номер и так далее.

- **Параллельный** — для кандидата с несколькими номерами при неуспешном дозвоне на первый номер, следующий звонок будет производиться на второй номер и так далее. После чего будет сформирована очередь перезвонов.
Для корректной работы параллельной стратегии по нескольким кандидатам интенсивность обзвона должна превышать один звонок в секунду.
- Включите нужный сценарий перезвона:
 - **Ошибка вызова** — оператор отклонил вызов из-за высокой нагрузки на линии / вызов с использованием некорректного CallerID / номер не существует или указан неверно.
 - **Не отвечен** — клиент не берет трубку.
 - Не результативен — клиент ответил, но продолжительность звонка была меньше установленной минимальной длительности.
 - **Занято** — линия занята.
 - **Ответил автоответчик** — на звонок ответил автоответчик. Для использования этой функции в задании на обзвон должна быть включена Система определения «Человек/робот». Она включается во вкладке Общее.
- (Опционально) Настройте **Общие ограничения**:
 - **Перезвонов кандидату**: укажите максимальное количество вызовов кандидату.
 - **Перезвонов по номеру**: укажите максимальное количество вызовов по номеру телефона кандидата.
- Укажите максимальное количество попыток перезвона клиенту. Стратегии перезвонов включают в себя первую попытку дозвона до кандидата.
- Укажите интервал, через который будет выполнен следующий вызов.
Подробнее ознакомиться с работой правил перезвона можно на примере ниже.

Пример работы правил перезвона:

Если при настройке задания на обзвон вы установили флажок для параметра Использовать стратегию перезвона, затем выбрали правило Не отвечен и указали Звонить не более 3-х раз с интервалом 60 сек, то работать оно будет следующим образом:

- Если первый звонок не получил ответа, будет совершено всего 3 звонка: первичный звонок и 2 перезвона по правилу Не отвечен. Каждая повторная попытка будет выполнена через 60 секунд после завершения предыдущей. То есть, через 60 секунд после завершения вызова система предпримет первую попытку перезвона. Если клиент снова не ответит, то через 60 секунд после завершения первой попытки система предпримет вторую попытку перезвона. Если при выполнении второй попытки система дозвонится до клиента или клиент снова не возьмет трубку, то система больше не будет предпринимать дальнейших попыток дозвониться.
- Если первый звонок остается без ответа, но клиент принимает вызов при первой попытке перезвона, то система не будет предпринимать следующую попытку перезвона.

- Если после первого звонка не получен ответ, и при выполнении первой попытки перезвона система обнаружит другой статус (например, номер будет занят), то система будет продолжать пытаться дозвониться до тех пор, пока снова не обнаружит статус «Нет ответа». Иными словами, перезвон будет учтен только в том случае, если клиент ответит на звонок или система определит статус «Нет ответа». В такой ситуации система продолжит текущую попытку перезвона, и счетчик выполненных попыток перезвона для выбранного правила останется неизменным.
- Если выбраны правила Не отвечен и Не результативен одновременно, и вызов оказывается неуспешным с пропущенным первым звонком, а затем клиент, при попытке перезвона по правилу Не отвечен, отвечает на звонок, но завершает его до истечения установленной продолжительности, то система повторит попытку перезвона по правилу Не отвечен. При этом она учтет первую попытку перезвона по правилу Не результативен. Счетчик попыток перезвона изменится только для правила Не результативен, а для правила Не отвечен останется без изменений.

5. Нажмите кнопку **Сохранить**.

6.17. Добавление кандидатов

После создания задания на обзвон, в него необходимо добавить кандидатов на обзвон – это потенциальные клиенты, которым будет звонить бот.

Кандидатов можно добавить:

- Вручную по одному.
- Путем загрузки списка из файла.

Добавление вручную

1. Нажмите на строку задания на обзвон. Откроется окно **Звонки задания на обзвон**.
 2. Нажмите **Добавить кандидатов вручную** в центре страницы, если ни одного кандидата еще не было добавлено или  в шапке страницы.
 3. В выпадающем списке выберите **Добавить кандидатов вручную**. Откроется окно **Добавление кандидатов на обзвон для задания**.
 4. Добавьте кандидатов. Каждый кандидат добавляется отдельно.
 - Кандидат 1:
 - **Номер**: введите номер телефона первого кандидата. Для каждого кандидата можно добавить несколько номеров. Для этого нажмите **+Добавить в поле ниже**. При необходимости, вы можете удалить номер нажав .
 - Кандидат 2:
 - Нажмите **+Добавить кандидата**.
 - Добавьте номер телефона, как описано выше.
- Последующих кандидатов добавляйте таким же образом.

5. (Опционально) Включите **Включить проверку номеров**, чтобы система проверила корректность формата номера при добавлении кандидата.
6. Нажмите **Сохранить**. Кандидаты будут добавлены в задание на обзвон.

Загрузка кандидатов из файла

1. Нажмите на строку задания на обзвон. Откроется окно **Звонки задания на обзвон**.
2. Нажмите  → **Скачать шаблон файла загрузки**. Начнется загрузка файла на ваш компьютер.
3. Откройте скачанный файл, чтобы составить таблицу с данными всех кандидатов. Структура таблицы является строго определенной, ее изменение может привести к ошибкам обработки файла. Для каждого нового кандидата информация вносится с новой строки. Данные записываются следующим образом:
 - **Номер телефона**: укажите номер телефона кандидата.
 - (Опционально) **Внешний идентификатор клиента**: внесите номер клиента во внешней системе, который позволит его идентифицировать. Задается в произвольном формате, используется любая последовательность символов. Если оставить пустым, то после загрузки файла система автоматически установит произвольное значение.
 - (Опционально) **Часовой пояс**: внесите **часовой пояс** клиента. Используется в случае, если часовой пояс не совпадает с регионом номера телефона.
 - Значения от -12 до 12 определяются как часы таймзоны UTC.
 - Значения от -3599 до -13 определяются как -1 час таймзоны UTC.
 - Значения от 13 до 3599 определяются как +1 час таймзоны UTC.
 - Значения выше 3600 считаются секундами таймзоны UTC и округляются до ближайшего целого часа, граничные значения округляются в большую сторону. Например, если указать «9000», то система определит это значение как секунды и округлит до ближайшего часа в большую сторону (т.е. = 10800).
 - Значения выходящие за рамки от -43200 до 43200, а также нечисловые значения (в том числе незаполненное поле) учитываться не будут, в этом случае часовой пояс будут определяться по номеру телефона кандидата. Если часовой пояс по номеру телефона определить не удалось, то система определит его как UTC+0.
 - (Опционально) Внесите собственные пользовательские параметры (переменные), начиная с 4 столбца, для дальнейшего использования их в сценариях. Количество параметров не ограничено. Наименование каждой новой переменной задается заголовком в отдельном столбце без фигурных скобок (Имя, Город и т.д.). Значение переменной задается в строке клиента (*Александр, Уфа* и т.д.).
4. Загрузите файл с кандидатами:

- Нажмите  → **Загрузить кандидатов из файла.**
 - Выберите ранее составленный файл. Кандидаты на обзвон будут загружены в список из файла.
5. (Опционально) Если вы указали в файле пользовательские переменные, то необходимо добавить их в сценарий. Для этого:
- Перейдите в раздел **Сценарии ботов.**
 - В списке сценариев выберите тот, что добавлен в задание на обзвон. Откроется редактор сценариев.
 - Выберите стартовый блок. Справа откроется окно настроек.
 - Перейдите на вкладку **Переменные и шаблоны.**
 - Заполните переменные, которые будут использоваться в сценарии:
 - Нажмите **+Добавить.**
 - **Название:** укажите название переменной, т.е. имена столбцов пользовательских параметров из файла с кандидатами. Названия переменных и названия столбцов должны быть идентичны.
 - **Тип:** укажите тип данных: например, для имени – Имя или отчество, для города – Текстовые данные и т.д.
 - Повторите действия для всех переменных. Каждая переменная добавляется в новую строку.
 - Нажмите **Сохранить** в шапке страницы.

После добавления и настройки кандидатов вы сможете [запустить обзвон.](#)

Исключение кандидата из обзвона

При необходимости исключения кандидата из обзвона выполните следующее:

1. Перейдите в раздел **Голосовые боты.**
2. Нажмите на строку задания на обзвон. Откроется окно **Звонки задания на обзвон.**
3. Во вкладке **Кандидаты** наведите курсор на строку кандидата и нажмите .

6.18. Управление обзвоном

6.18.1. Запуск обзвона

Запуск обзвона может производиться как вручную, так и автоматически в указанное время. Режим запуска определяется при [создании задания на обзвон.](#)

Вручную

Если при создании задания на обзвон вы выбрали режим запуска задания Вручную, выполните следующие действия:

1. Перейдите в раздел **Голосовые боты.**
2. Выберите задание на обзвон, которое хотите запустить.
3. В строке задания нажмите .

Автоматически

Если при создании задания на обзвон вы выбрали режим запуска задания В указанное время, то обзвон начнется автоматически в то время, которое вы указали в задании.

6.18.2. Пауза

1. В разделе **Голосовые боты** нажмите на строку задания на обзвон. Откроется страница **Звонки задания на обзвон**.
2. Нажмите кнопку **Пауза** в шапке страницы.
3. Чтобы продолжить обзвон, поставленный на паузу, нажмите кнопку **Продолжить обзвон**.

6.18.3. Отмена обзвона

1. В разделе **Голосовые боты** нажмите на строку задания на обзвон. Откроется страница **Звонки задания на обзвон**.
2. Нажмите кнопку **Отменить обзвон** в шапке страницы.
3. Во всплывающем окне подтвердите отмену обзвона. Обзвон будет остановлен. Возобновить обзвон после отмены невозможно.

6.18.4. Перезапуск вызовов

1. В разделе **Голосовые боты** нажмите на строку задания на обзвон. Откроется страница **Звонки задания на обзвон**.
2. Нажмите .
3. В выпадающем списке выберите вызовы, которые необходимо запустить заново:
 - Вызовы с ошибкой.
 - Неуспешные вызовы — это вызовы, которые остались без ответа.

6.19. Работа со звонками в задании на обзвон

После запуска задания на обзвон на вкладке **Звонки** вы можете:

- смотреть список совершенных звонков;
- копировать ID звонка для обращения в техническую поддержку;
- получать подробную информацию о звонках;
- прослушивать запись звонка;
- переходить в редактор сценариев.

6.19.1. Просмотр списка звонков

1. Перейдите в раздел **Голосовые боты** [личного кабинета](#).
2. Во вкладке **Задания** нажмите в строке с нужным заданием. Откроется страница **Звонки задания на обзвон**.

3. Перейдите на вкладку **Звонки**. Откроется вкладка со списком всех звонков этого задания.

6.19.2. Копирование ID звонка

Копирование ID может быть полезно, например, для передачи его в техническую поддержку.

Чтобы скопировать ID:

1. В разделе **Голосовые боты** → **Задания** нажмите на нужное задание на обзвон.
2. Перейдите на вкладку **Звонки**.
3. Наведите курсор на строку звонка.
4. Нажмите на . После этого идентификатор будет скопирован в буфер обмена на вашем компьютере.

6.19.3. Получение подробной информации о звонке

1. Нажмите на строку с нужным звонком. Появится окно с подробной информация о вызове. Вкладка **Информация** содержит подробности о вызове:
 - **Ссылка на задание** – ссылка на задание на обзвон в личном кабинете.
 - **Скрипт** – ссылка на сценарий бота, который участвует в обзвоне.
 - **ID** – идентификатор вызова. Может быть скопирован в буфер обмена нажатием на .
 - **Телефон кандидата** – телефонный номер клиента, по которому был совершен звонок.
 - **Статус** – статус проведенного звонка: Набор номера, Успешный, Абонент занят, Телефонная линия перегружена и т.д.
 - **Дата и время звонка** – дата и время начала звонка (заполняется, если звонок был успешен).
 - **Длительность** – продолжительность звонка в секундах.
 - **Запуск** — статус звонка, указывающий на то, был ли звонок запущен вручную или автоматически.
 - **Стоимость** – стоимость звонка в рублях.
2. Перейдите на вкладку **Расшифровка**. Вы увидите текстовую расшифровку звонка.
3. Нажмите **Заккрыть**.

 **Максимальный срок хранения данных о звонке — 3 месяца.**

6.19.4. Прослушивание аудиозаписи звонка

1. Наведите курсор на строку нужного звонка.
2. Нажмите на появившуюся кнопку воспроизведения .

3. (Опционально) Поставьте аудиозапись на паузу, нажав .

6.19.5. Переход в редактор сценариев

1. Наведите курсор на строку звонка.
2. Нажмите на .
3. В выпадающем списке нажмите **Редактировать бота**. Откроется редактор сценария с тем сценарием, который использовался при обзвоне.

6.20. Управление шаблонами

Шаблоны заданий на обзвон позволяют:

1. Выполнять обзвоны клиентов с помощью голосового бота во внешних CRM-системах (например, Битрикс24, amoCRM).
2. Создавать задания для обзвона быстро: нужно только указать название задания, а все остальные параметры будут автоматически взяты из шаблона.

Управление шаблонами происходит в [личном кабинете](#) в разделе Голосовые боты на вкладке Шаблоны заданий.

Здесь вы можете:

- Создавать новые шаблоны заданий на обзвон.
- Создавать задания на обзвон с помощью шаблона.
- Редактировать ранее созданные шаблоны.
- Удалять шаблоны, которые больше не нужны.

6.20.1. Создание шаблона задания на обзвон

1. Перейдите в раздел **Голосовые боты** [личного кабинета](#).
2. Перейдите на вкладку **Шаблоны заданий**.
3. Нажмите **Создать шаблон задания** или  в правом верхнем углу. Откроется окно **Создание шаблона задания**.
4. Выполните шаги по настройке шаблона, которые идентичны шагам по [созданию задания на обзвон](#).

6.20.2. Редактирование шаблона задания на обзвон

1. Перейдите в раздел **Голосовые боты** → **Шаблоны заданий**. Откроется страница со списком ранее созданных шаблонов.
2. Наведите курсор на нужный шаблон и нажмите на  в строке шаблона справа. Откроется окно **Редактирование шаблона задания** с параметрами шаблона.
3. Внесите необходимые изменения. Действия по редактированию шаблона аналогичны тем, которые описаны в статье [Создание задания на обзвон](#).
4. Нажмите Сохранить внизу окна.

6.20.3. Удаление шаблона задания на обзвон

1. Перейдите в раздел **Голосовые боты** → **Шаблоны заданий**. Откроется страница со списком ранее созданных шаблонов.
2. Наведите курсор на нужный шаблон.
3. Нажмите на  в строке шаблона справа.
4. Подтвердите удаление, нажав кнопку **Удалить**. Выбранный шаблон будет удален.

6.21. Поиск аудиозвонка по номеру телефона

Вариант 1. Поиск звонка в задании на обзвон.

Чтобы найти звонок по номеру телефона в задании на обзвон выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** [личного кабинета](#).
2. Во вкладке **Задания** нажмите в строке с нужным заданием. Откроется страница **Звонки задания на обзвон**.
3. Перейдите на вкладку **Звонки**. Откроется вкладка со списком всех звонков этого задания.
4. В поле **Поиск** введите номер телефона, содержащий интересующий вас звонок.

Вариант 2. Поиск звонка из всех звонков за определенный период.

Чтобы найти конкретный звонок по номеру телефона выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Детализация**. Выберите вкладку **Входящие** или **Исходящие** для просмотра информации по звонкам.
2. В поле **Дата** нажмите на . Откроется календарь.
3. Укажите период времени, за который вы хотите получить отчет. По умолчанию установлен период с первого числа текущего месяца по сегодняшней день.
4. (Опционально) В поле **Сценарий** из выпадающего списка выберите конкретного бота, диалоги с которым вы хотите получить. Если поле оставить пустым, то сформируется отчет диалогов всех ботов в личном кабинете за указанный период.
5. Обратитесь к столбцам вкладки, где указаны номера телефонов, чтобы найти интересующий вас звонок. Во вкладке **Входящие** номера телефонов указаны в столбцах **Вызывающий номер** и **Телефон**. Во вкладке **Исходящие** номера телефонов указаны в столбце **Вызывающий номер**.

6.22. Просмотр входящих и исходящих звонков

Вариант 1. Чтобы просмотреть звонки из задания на обзвон выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** [личного кабинета](#).
2. Во вкладке **Задания** нажмите в строке с нужным заданием. Откроется страница **Звонки задания на обзвон**.
3. Перейдите на вкладку **Звонки**. Откроется вкладка со списком всех звонков этого задания.

Вариант 2. Чтобы просмотреть все звонки за определенный период выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Детализация**. Выберите вкладку **Входящие** или **Исходящие** для просмотра информации по звонкам.
2. В поле **Дата** нажмите на . Откроется календарь.
3. Укажите период времени, за который вы хотите получить отчет. По умолчанию установлен период с первого числа текущего месяца по сегодняшний день.
4. (Опционально) В поле **Сценарий** из выпадающего списка выберите конкретного бота, диалоги с которым вы хотите получить. Если поле оставить пустым, то сформируется отчет диалогов всех ботов в личном кабинете за указанный период.

6.23. Добавление оператора связи

Операторы связи используются для принятия входящих и совершения исходящих вызовов. Вы можете добавить в ваш кабинет несколько операторов связи. Если вы используете АТС TWIN, то настраивать операторов связи не нужно.

Если в процессе настройки у вас возникли вопросы, вы можете обратиться за консультацией в [техническую поддержку](#) или заказать платную [услугу по настройке](#).

Чтобы добавить нового оператора связи, выполните следующее:

1. Перейдите в [личный кабинет](#).
2. Перейдите в раздел **Голосовые боты** → **Настройки**.
3. Перейдите на вкладку **Операторы**. Откроется страница со списком операторов связи.
4. Нажмите  в верхнем левом углу страницы или **Подключение оператора**, если еще ни один оператор подключен не был. Откроется окно **Подключение оператора**.
5. Заполните поля во вкладке **Общее**:
 - **Наименование**: введите название оператора связи.
 - (Опционально) **Комментарий**: введите дополнительную информацию о новом операторе.

- **SRC порт для отправки вызовов:** введите порт платформы. Нужен для передачи информации и для установления соединения между абонентами и его поддержания (т.е. данных, циркулирующих через SIP-протокол). По умолчанию используется порт 5060.
- **Режим DTMF:** из выпадающего списка выберите режим DTMF (двухтональный многочастотный сигнал) — метод передачи сигналов набора номера, который используется в телефонной связи. Чаще всего используется RFC2833/4733.
- **Основной адрес подключения:** введите IP-адрес или доменное имя с портом (например, 192.168.1.2:5062). Используйте протокол UDP. Чтобы добавить дополнительную точку подключения, нажмите кнопку **+Добавить**. Заполните появившееся поле.

6. (Опционально) Выполните настройки во вкладке **Авторизация:**

- **Авторизация по логину и паролю:** активируйте переключатель, чтобы авторизация проходила по логину и паролю. В противном случае, авторизации будут проходить по IP-адресу (IP2IP).
- **При необходимости добавлять FromUser:** установите флажок, чтобы добавить параметр **FromUser** при создании транка с телефонией.
- **Для подключения требуется Authuser:** установите флажок, чтобы добавить обязательное поле **Authuser**.
- **Для подключения требуется Fromdomain:** установите флажок, чтобы добавить обязательное поле **Fromdomain**.
- **Логин:** введите логин для подключения.
- **Authuser:** введите имя пользователя для авторизации на SIP сервере.
- **Fromdomain:** введите имя домена, используемое в заголовке поля From в запросах к этому транку.
- **Пароль:** введите пароль для подключения.
- **Для звонков необходима регистрация:** установите флажок, если требуется активная регистрация на удаленной стороне. Если флажок не установлен, то при исходящем звонке система будет проходить только процедуру авторизации.

7. Выполните настройки во вкладке **Назначение транка:**

- (Опционально) **Принятие входящих вызовов:** активируйте переключатель, чтобы принимать входящие вызовы.
 - (Опционально) **Модификация CallerID:** в случае, если при входящем звонке определяемый номер не соответствует стандарту, скорректируйте его:
 - **Удалить:** введите количество символов, которые необходимо удалить с начала определяемого номера.
 - **Добавить:** укажите символы, которые необходимо добавить в начало определяемого номера после удаления.
- (Опционально) **Совершение исходящих вызовов:** активируйте переключатель, если хотите совершать исходящие вызовы.
 - (Опционально) **Использовать в сервисах:** установите флажки напротив тех сервисов, в которых хотите использовать транк.

Если флажки не установлены, транк разрешено использовать во всех сервисах.

- (Опционально) **Модификация номера назначения:** скорректируйте номер назначения, передаваемый в транк, в соответствии с требованиями получателя вызова:
 - **Удалить:** введите количество символов, которые необходимо удалить с начала определяемого номера.
 - **Добавить:** укажите символы, которые необходимо добавить в начало определяемого номера.
 - (Опционально) **Модификация CallerID:** в случае, если при входящем звонке определяемый номер не соответствует стандарту, скорректируйте его:
 - **Удалить:** введите количество символов, которые необходимо удалить с начала определяемого номера.
 - **Добавить:** укажите символы, которые необходимо добавить в начало определяемого номера.
 - **Пропускная способность:** введите значения для следующих параметров:
 - **CPS:** максимально допустимое количество вызовов, инициируемых за одну секунду.
 - **Линии:** максимальное количество одновременных разговоров.
8. (Опционально) Настройте управление заголовками во вкладке **Заголовки**. Действие доступно только, если активировано **Совершение исходящих вызовов** в назначении транка.
9. После заполнения всех необходимых параметров, нажмите **Сохранить**. Новый оператор появится в списке.

6.24. Добавление Caller ID внешней телефонии

Caller ID (CID) — это технология, которая автоматически определяет номер телефона абонента во время входящего вызова. Эта функция позволяет узнавать, от кого поступил вызов. Caller ID и АОН (Автоматический Определитель Номера), в контексте платформы TWIN, имеет одно значение.

Для того чтобы совершать обзвоны клиентов, необходимо иметь Caller ID. Вы можете приобрести Caller ID, оставив заявку в [техническую поддержку](#), или добавить имеющийся. Добавленный номер телефона будет отображаться на устройстве клиента при входящих вызовах.

Чтобы добавить Caller ID:

1. Перейдите в раздел **Голосовые боты** → **Настройки** личного кабинета.
2. Нажмите **Создать правила входящего звонка** или  в левом верхнем углу.

3. **Название номера:** введите название, которое будет отображаться в списке при выборе номера для обзвона.
4. **Номер телефона:** укажите номер телефона, который был предоставлен поставщиком.
5. (Опционально) **Комментарий:** При необходимости добавьте дополнительную информацию о добавляемом номере.
6. (Опционально) **Адрес вебхука:** нажмите **Добавить** и укажите адрес вебхука, на который будет отправляться информация о совершенных вызовах.
7. Настройте блок **Настройки** подключения:
 - **Транк** — код оператора связи. Выберите из выпадающего списка транк, через который будут совершаться вызовы.
 - **DID совпадает с номером:** выберите эту опцию, если номер телефона совпадает с DID. По умолчанию эта опция отключена. В случае если DID не совпадает с номером, отключите опцию и введите правильный номер в соответствующем поле ниже.
 - **Принудительно отправлять звонки через данное подключение:** выберите данную опцию, если номер используется в качестве АОНа, для обеспечения гарантированных звонков через выбранный транк, при игнорировании других правил маршрутизации.
8. Нажмите **Сохранить**.

6.25. Обработка входящих вызовов

Для приема входящих вызовов от клиентов вам необходимо добавить и настроить номер для обработки входящих звонков.

Подготовительные шаги

1. Перейдите в личный кабинет.
2. Добавьте [оператора связи](#).
3. Перейдите в раздел **Сценарии ботов**.
4. Создайте [бота](#), который будет отвечать на входящие вызовы.
5. Выполните первые 7 шагов, которые описаны в статье [Добавление CallerID внешней телефонии](#).

Настройка номера

1. Перейдите на вкладку **Вызовы**.
2. Включите переключатель **Направлять входящие вызовы**.
3. В поле **Бот** из выпадающего списка выберите сценарий, который был ранее создан.
4. Нажмите **Сохранить**.

6.26. Экспорт отчета по заданию на обзвон

После выполнения задания на обзвон вы можете получить по нему статистику. Для этого необходимо сформировать отчет по заданию на обзвон.

Обычный отчет

В этом отчете содержатся данные о каждом звонке, включая статусы, часовой пояс, регион, стоимость и многое другое. Этот отчет помогает оценить эффективность обзвонков и взаимодействие с клиентами, а также анализировать данные для улучшения процесса обзвона.

Чтобы сформировать отчет, выполните следующие шаги:

1. В разделе **Голосовые боты** нажмите на строку нужного задания на обзвон. Откроется страница **Звонки задания на обзвон**.
2. На вкладке **Кандидаты** нажмите на . Система сформирует отчет с подробной информацией о задании в виде файла в формате xlsx. Отчет сформируется в разделе **Отчеты**. На экране отобразится уведомление об этом.
3. Нажмите **Перейти в уведомлении** или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
4. Скачайте отчет нажав .

Расширенные отчеты

Отчет с произвольными результатами

Это один из расширенных отчетов по заданию на обзвон. В последних столбцах содержит переменные и значения, зафиксированные с помощью блока **Результат**.

Чтобы вывести переменные в отчет, в сценарии необходимо установить блок **Результат** с типом действия **Произвольный результат**, и зафиксировать переменные в поля **Переменные результата**.

Названия столбцов в отчете соответствуют названиям полей, заданным при сохранении в произвольный результат.

Чтобы сформировать отчет выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Задания**.
2. Нажмите на строку нужного задания на обзвон. Откроется страница **Звонки задания на обзвон**.
3. На вкладке **Кандидаты** нажмите на .
4. Из выпадающего списка выберите **Отчет с произвольными результатами**. Откроется окно **Формирование отчета с произвольными результатами**.

5. Выберите какие именно результаты необходимо вывести в отчет.
6. Нажмите **Готово**. Система сформирует отчет в виде файла в формате `xlsx`. Отчет сформируется в разделе **Отчеты**. На экране отобразится уведомление об этом.
7. Нажмите **Перейти** в уведомлении или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
8. Скачайте отчет нажав .

Отчет с переменными

Это один из расширенных отчетов по заданию на обзвон. В последних столбцах отчет содержит исходные значения переменных, которые были переданы на старт звонка вместе с кандидатом.

Исходные значения переменных попадут в отчет даже в случае их изменений в ходе разговора.

Названия столбцов имеют вид `variables-name`, где `name` – название переменной.

Чтобы сформировать отчет выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Задания**.
2. Нажмите на строку нужного задания на обзвон. Откроется страница **Звонки задания на обзвон**.
3. На вкладке **Кандидаты** нажмите на .
4. В выпадающем списке выберите **Отчет с переменными**. Система сформирует отчет в виде файла в формате `xlsx`. Отчет сформируется в разделе **Отчеты**. На экране отобразится уведомление об этом.
5. Нажмите **Перейти в уведомлении** или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
6. Скачайте отчет нажав .

6.27. Экспорт отчета по исходящим вызовам

Этот отчет предоставляет подробную информацию об исходящих звонках за выбранный период времени. В нем содержатся данные о каждом звонке, включая статусы, стоимость, а также информацию о голосовых роботах и другое. Отчет помогает оценить эффективность обзвонков и взаимодействие с клиентами, а также анализировать данные для улучшения процесса обзвона.

Чтобы сформировать отчет, выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Детализация**. Откроется страница с информацией по звонкам.
2. Перейдите на вкладку **Исходящие**.
3. В поле **Дата** нажмите на . Откроется календарь.
4. Укажите период времени, за который вы хотите получить отчет. По умолчанию установлен период с первого числа текущего месяца по сегодняшний день.
5. (Опционально) В поле **Сценарий** из выпадающего списка выберите конкретного бота, диалоги с которым вы хотите получить. Если поле оставить пустым, то сформируется отчет диалогов всех ботов в личном кабинете за указанный период.
6. Нажмите кнопку **Отчет**. В правом нижнем углу появится уведомление о том, что отчет добавлен в очередь.
7. Нажмите **Перейти** в уведомлении или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
8. Скачайте отчет нажав .

6.28. Экспорт отчета по входящим вызовам

Этот отчет предоставляет подробную информацию о входящих звонках за выбранный период времени. В нем содержатся данные о каждом звонке, включая статусы, стоимость, а также информация о голосовых роботах и многое другое. Отчет помогает оценить эффективность взаимодействия с клиентами.

Чтобы сформировать отчет выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Детализация**. Откроется вкладка **Входящие** с информацией по звонкам.
2. В поле **Дата** нажмите на . Откроется календарь.
3. Укажите период времени, за который вы хотите получить отчет. По умолчанию установлен период с первого числа текущего месяца по сегодняшний день.
4. (Опционально) В поле **Сценарий** из выпадающего списка выберите конкретного бота, диалоги с которым вы хотите получить. Если поле оставить пустым, то сформируется отчет диалогов всех ботов в личном кабинете за указанный период.
5. Нажмите кнопку **Отчет**. В правом нижнем углу появится уведомление о том, что отчет добавлен в очередь.
6. Нажмите **Перейти** в уведомлении или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
7. Скачайте отчет нажав .

6.29. Экспорт отчета по телефонии с произвольными результатами

Отчет используется для аналитики. В последних столбцах содержит данные, зафиксированные с помощью блока Результат.

Подготовительные шаги

1. Перейдите в раздел Сценарии ботов личного кабинета.
2. Создайте новый сценарий или выберите нужный из ранее созданных.
3. В сценарии установите блок Результат с типом действия Произвольный результат.

Фиксация результата будет передана в отчет, только если бот попадет в этот блок.

4. Укажите названия и значения переменных настройках блока в разделе Переменные результата в полях Название и Значение . Добавляются поля по кнопке Добавить, а удаляются по клику на .

Может быть добавлено любое количество переменных.

5. Перейдите в раздел [Голосовые боты](#) → [Задания](#).
6. [Создайте](#) задание на обзвон, используя ранее настроенный сценарий.
7. [Добавьте](#) в задание кандидатов и [запустите](#) обзвон.

Экспорт отчета

После выполнения задания на обзвон сформируйте отчет с произвольными результатами. Для этого:

1. Перейдите в раздел [Голосовые боты](#) → [Задания](#). Откроется страница со списком всех заданий на обзвон.
2. Нажмите на строку нужного задания на обзвон. Откроется страница **Звонки задания на обзвон**.
3. На вкладке **Кандидаты** нажмите на .
4. Из выпадающего списка выберите **Отчет с произвольными результатами**. Откроется окно Формирование отчета с произвольными результатами со списком всех указанных переменных.
5. (Опционально) Удалите переменные, которые не нужно выгружать в отчет. Для этого нажмите на  в строке выбранной переменной. Чтобы восстановить случайно удаленную переменную, нажмите **Добавить** и введите ее название.
6. Нажмите **Готово**. В правом нижнем углу появится уведомление о том, что отчет добавлен в очередь. Система сформирует отчет в виде файла в формате `xlsx`. В сформированном отчете столбцы произвольного результата будут отображаться в случайном порядке и иметь название в формате `result-name`, где `name` — название переменной, заданной при сохранении в

произвольный результат. После формирования отчета в правом нижнем углу появится уведомление об успехе.

7. Нажмите **Скачать** в уведомлении, чтобы загрузить сформированный отчет на ваш компьютер или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка. Нажмите на  в строке нужного отчета.

6.30. Выгрузка входящих звонков

Выгрузка входящих звонков формируется в виде отчета. Чтобы сформировать отчет выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Детализация**. Откроется вкладка **Входящие** с информацией по звонкам.
2. В поле **Дата** нажмите на . Откроется календарь.
3. Укажите период времени, за который вы хотите получить отчет. По умолчанию установлен период с первого числа текущего месяца по сегодняшний день.
4. (Опционально) В поле **Сценарий** из выпадающего списка выберите конкретного бота, диалоги с которым вы хотите получить. Если поле оставить пустым, то сформируется отчет диалогов всех ботов в личном кабинете за указанный период.
5. Нажмите кнопку **Отчет**. В правом нижнем углу появится уведомление о том, что отчет добавлен в очередь.
6. Нажмите **Перейти** в уведомлении или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
7. Скачайте отчет нажав .

6.31. Выгрузка исходящих звонков

Выгрузка исходящих звонков формируется в виде отчета. Чтобы сформировать отчет, выполните следующие шаги:

1. Перейдите в раздел **Голосовые боты** → **Детализация**. Откроется страница с информацией по звонкам.
2. Перейдите на вкладку **Исходящие**.
3. В поле **Дата** нажмите на . Откроется календарь.
4. Укажите период времени, за который вы хотите получить отчет. По умолчанию установлен период с первого числа текущего месяца по сегодняшний день.

5. (Опционально) В поле **Сценарий** из выпадающего списка выберите конкретного бота, диалоги с которым вы хотите получить. Если поле оставить пустым, то сформируется отчет диалогов всех ботов в личном кабинете за указанный период.
6. Нажмите кнопку **Отчет**. В правом нижнем углу появится уведомление о том, что отчет добавлен в очередь.
7. Нажмите **Перейти** в уведомлении или перейдите в раздел **Отчеты**, который содержит список всех ранее сформированных отчетов. Только что созданный отчет будет отображаться вверху списка.
8. Скачайте отчет нажав .

6.32. Начало работы с BPL

BPL (Bot Programming Language) — это язык выражений, специально разработанный для создания, настройки и управления логикой чат-ботов. Он позволяет легко описывать сложные сценарии поведения ботов с использованием понятного и лаконичного синтаксиса.

BPL применяется для управления:

- Обработкой входящих сообщений.
- Выполнением условий и ветвлений.
- Управлением данными, такими как переменные, массивы и строки.
- Вызовом встроенных функций и взаимодействием с внешними API.

BPL является связующим звеном между пользователем и логикой бота. Он позволяет:

- Описывать реакции бота на сообщения.
- Создавать сложные сценарии диалога.
- Управлять состоянием и памятью бота.
- Реализовывать динамические и адаптивные реакции на запросы.

Преимущества:

1. Мощные функции, включая:
2. Простой и интуитивный синтаксис. BPL разработан с акцентом на читабельность и минимализм, что делает его доступным как для опытных программистов, так и для новичков.
3. Гибкость и расширяемость. BPL позволяет:
 - Легко обновлять и изменять логику бота.
 - Интегрировать пользовательские функции и библиотеки.
 - Адаптироваться под уникальные сценарии использования.
4. Быстрая разработка благодаря своему лаконичному синтаксису.

6.33. Основные понятия BPL

6.33.1. Операции

Операции — это основа для выполнения вычислений, логики и обработки данных в языке выражений бота. Они позволяют складывать числа, проверять условия, манипулировать строками, работать с коллекциями и многое другое.

Приоритет операций

Приоритет определяет порядок выполнения операций. Операции с более высоким приоритетом выполняются раньше. Чтобы явно указать порядок выполнения, используйте круглые скобки.

Таблица приоритетов операций:

Операция	Приоритет	Ассоциативность	Описание
**	12	Правая	Возведение в степень
+, - (унарные)	11	—	Унарные плюс и минус
~	11	—	Побитовая инверсия
!	10	—	Логическое отрицание
*, /, \, %	9	Левая	Умножение, деление, остаток
+, -	8	Левая	Сложение, вычитание
::	8	Левая	Конкатенация строк
<, <=, >, >=	7	—	Сравнения

<code>==, !=</code>	6	–	Равенство, неравенство
<code>&</code>	5	Левая	Побитовое И
<code>^</code>	4	Левая	Побитовое исключающее ИЛИ
<code> </code>	3	Левая	Побитовое ИЛИ
<code>&&</code>	2	Левая	Логическое И
<code> </code>	1	Левая	Логическое ИЛИ
Присваивание (<code>=</code> и др.)	0	Правая	Операции присваивания

Арифметические операции

Арифметические операции выполняются над числами. Если один из операндов не является числом, интерпретатор попытается преобразовать его в число.

Операция	Описание	Пример
<code>+\$a</code>	Преобразование в число	<code>\$a = +"5"</code>
<code>-\$a</code>	Смена знака	<code>\$a = -5</code>
<code>\$a + \$b</code>	Сложение	<code>\$c = \$a + \$b</code>
<code>\$a - \$b</code>	Вычитание	<code>\$c = \$a - \$b</code>

<code>\$a * \$b</code>	Умножение	<code>\$c = \$a * \$b</code>
<code>\$a / \$b</code>	Деление	<code>\$c = \$a / \$b</code>
<code>\$a \ \$b</code>	Целочисленное деление	<code>\$c = \$a \ \$b</code>
<code>\$a % \$b</code>	Остаток от деления	<code>\$c = \$a % \$b</code>
<code>\$a ** \$b</code>	Возведение в степень	<code>\$c = \$a ** \$b</code>

Логические операции

Логические операции работают с булевыми значениями. Если операнд не является булевым, он будет преобразован.

Операция	Описание	Пример
<code>!\$a</code>	Логическое НЕ	<code>\$b = !\$a</code>
<code>\$a && \$b</code>	Логическое И	<code>\$c = \$a && \$b</code>
<code>\$a \$b</code>	Логическое ИЛИ	<code>\$c = \$a \$b</code>

Таблица преобразования в булевы значения:

Тип	Значение	Булево значение
<code>nil</code>	<code>nil</code>	<code>false</code>

Число	0	false
Число	Ненулевое	true
Строка	Пустая	false
Строка	Непустая	true
Объект	Любой	true

Побитовые операции

Побитовые операции работают с битовыми представлениями чисел.

Операция	Описание	Пример
$\sim \$a$	Побитовая инверсия	$\$b = \sim \a
$\$a \& \b	Побитовое И	$\$c = \$a \& \$b$
$\$a \b	Побитовое ИЛИ	$\$c = \$a \$b$
$\$a \wedge \b	Исключающее ИЛИ	$\$c = \$a \wedge \$b$

Операции со строками

Строки можно объединять с помощью операции конкатенации `::`.

Операция	Описание	Пример
----------	----------	--------

```
"a" :: "b"
```

Объединение строк

```
"a" :: "b" == "ab"
```

Преобразование других типов данных в строки:

Тип	Значение	Строковое представление
<code>nil</code>	<code>nil</code>	<code>" "</code>
Число	<code>123.45</code>	<code>"123.45"</code>
Логический	<code>true</code>	<code>"true"</code>
Логический	<code>false</code>	<code>"false"</code>
Объект	Любой	Сериализация

Операции сравнения

Сравнения выполняются над любыми типами данных, возвращая булевы значения.

Операция	Описание	Пример
<code>\$a == \$b</code>	Равенство	<code>\$a == \$b</code>
<code>\$a != \$b</code>	Неравенство	<code>\$a != \$b</code>
<code>\$a > \$b</code>	Больше	<code>\$a > \$b</code>

<code>\$a >= \$b</code>	Больше или равно	<code>\$a >= \$b</code>
<code>\$a < \$b</code>	Меньше	<code>\$a < \$b</code>
<code>\$a <= \$b</code>	Меньше или равно	<code>\$a <= \$b</code>

Операции с коллекциями

Коллекции поддерживают операцию объединения.

Операция	Описание	Пример
<code>(1, 2) + (3, 4)</code>	Объединение кортежей	<code>(1, 2, 3, 4)</code>
<code>['a'] + ['b']</code>	Объединение списков	<code>['a', 'b']</code>
<code>{'a': 0} + {'b': 1}</code>	Объединение словарей	<code>{'a': 0, 'b': 1}</code>

Операции присваивания

Присваивание позволяет установить значение переменной. Также доступны комбинированные операции, например, сложение с присваиванием.

Операция	Описание	Пример
<code>\$a = 123</code>	Присваивание	<code>\$a = 123</code>
<code>\$a += 1</code>	Сложение с присваиванием	<code>\$a = \$a + 1</code>
<code>\$a -= 1</code>	Вычитание с присваиванием	<code>\$a = \$a - 1</code>

```
$a *= 2
```

Умножение с присваиванием

```
$a = $a * 2
```

Пример множественного присваивания:

```
($x, $y, $z) = (1, 2, 3) // Присваиваем значения нескольким переменным
```

6.33.2. Типы данных

Работа с данными в сценариях бота основывается на различных типах данных. Это позволяет удобно организовать информацию и выполнять вычисления.

Основные типы данных

Числа

Все числа в системе представлены как вещественные (с плавающей точкой). Это означает, что они могут содержать дробные части. Минимальные и максимальные значения чисел зависят от среды, в которой выполняется интерпретатор.

Пример:

```
$x = 2.5 * 4 // Простой расчет с числами
```

Строки

Строки — это последовательности символов, заключенные в одинарные ' или двойные " кавычки.

Важное отличие: внутри двойных кавычек можно использовать специальные символы, которые интерпретируются особым образом:

Специальная последовательность	Описание
<code>\n</code>	Перевод строки
<code>\r</code>	Возврат каретки
<code>\t</code>	Горизонтальная табуляция
<code>\"</code>	Двойная кавычка

В одинарных кавычках символы трактуются буквально, за исключением последовательности `\'`, которая означает одинарную кавычку.

Пример:

```
say("Привет\n\"Медвед!\") // Двойные кавычки: поддержка  
специальных символов say('Привет\n\'Медвед!\') // Одинарные  
кавычки: всё трактуется как есть
```

Булевы значения

Булевы значения имеют всего два состояния: `true` (истина) и `false` (ложь). Они часто используются в условных выражениях.

Пример:

```
$x = true // $x содержит ИСТИНА $y = !$x // $y содержит ЛОЖЬ  
(инверсия значения)
```

Объекты

Объекты — это сложные структуры данных с встроенными свойствами и методами. Для доступа к свойствам и методам используется точка (`.`). Например, вы можете получить текст первого сообщения клиента из очереди с помощью объекта `ClientMessage`.

Пример:

```
$first = queue.first() // Получаем объект ClientMessage  
$firstMessage = $first.message // Доступ к свойству message
```

Также можно динамически вызывать методы объекта.

Пример:

```
// Случайный выбор метода $n = rand(0, 1) $method = ["first",  
"last"].get($n) // Вызов метода $firstOrLast = queue.$method
```

Коллекции данных

Кортежи

Кортеж — это упорядоченная и неизменяемая коллекция. Элементы кортежа можно извлекать, но нельзя изменять.

Создаются с помощью круглых скобок.

Пример:

```
$items = (1, 2, 3) // Кортеж из трех элементов $empty = () //
```

```
Пустой кортеж $single = ('a',) // Кортеж из одного элемента  
(запятая обязательна)
```

Методы работы с кортежами:

- `.count()` — количество элементов.
- `.first()` — первый элемент.
- `.last()` — последний элемент.
- `.get(index)` — элемент по индексу.

Пример:

```
$count = $items.count() // Количество элементов $first =  
$items.first() // Первый элемент
```

Списки

Список — это изменяемая коллекция, которую можно дополнять, изменять и очищать.

Создаются с помощью квадратных скобок.

Пример:

```
$items = [1, 2, 3] // Список из трех элементов $empty = [] //  
Пустой список
```

Методы работы со списками:

- `.append(item)` — добавить элемент в конец.
- `.prepend(item)` — добавить элемент в начало.
- `.get(index)` — элемент по индексу.
- `.clear()` — удалить все элементы.

Пример:

```
$items.append(4) // Добавить элемент $first = $items.get(0) //  
Получить первый элемент
```

Ассоциативные массивы (словари)

Словарь — это коллекция пар ключ-значение. Создаются с помощью фигурных скобок.

Пример:

```
$map = {'a': 1, 'b': 2} // Словарь с двумя парами $empty = {} //
```

Пустой словарь

Методы работы со словарями:

- `.get(key)` — значение по ключу.
- `.set(key, value)` — установить значение по ключу.
- `.keys()` — список всех ключей.
- `.values()` — список всех значений.

Пример:

```
$value = $map.get('a') // Получить значение по ключу $map.set('c', 3) // Добавить новую пару
```

Специальное значение: `nil`

`nil` — это специальное значение, обозначающее «ничего». Оно используется для неинициализированных переменных или для явного указания на отсутствие данных.

Пример:

```
$x = nil // Присвоение nil
```

6.33.3. Функции

Функции — это основные инструменты для выполнения сложных операций в сценариях бота. Они предоставляют готовые решения для задач, таких как работа с датами, строки, математические вычисления и другие операции.

Функцию можно вызвать по имени, передав необходимые параметры. Она обрабатывает эти параметры и возвращает результат.

Пример:

```
$number = math.rand(5, 10) // Вызов функции math.rand для получения случайного числа от 5 до 10
```

Где:

- `math` — название модуля.
- `rand` — название функции.

Чтобы использовать функцию, нужно:

1. Укажите имя функции.
2. Передайте в круглых скобках параметры через запятую.

Формат вызова:

```
модуль.функция(параметры)
```

Пример:

```
$result = math.sqrt(16) // $result = 4 // Вычисляем квадратный  
корень из числа 16
```

Динамические имена

Имена функций можно вычислять прямо во время выполнения сценария. Это позволяет гибко выбирать, какую функцию использовать.

Пример:

```
$funcIdx = math.rand(0, 1) // Получаем случайное число (0 или 1)  
для выбора функции $number = math.(['min', 'max'].get($funcIdx))(3,  
7) // Динамически выбираем и вызываем функцию min или max //  
Результат: либо минимальное, либо максимальное значение из 3 и 7
```

Основные возможности

Функции охватывают множество задач:

1. Работа с числами — сложные математические вычисления.

```
$rounded = math.round(3.14159, 2) // $rounded = 3.14 // Пример:  
округление числа
```

2. Обработка строк — работа с текстом.

```
// Пример: приведение строки к нижнему регистру $text =  
string.toLowerCase("Привет Мир!") // $text = "привет мир!"
```

3. Дата и время — операции с текущей датой.

```
// Пример: получение текущей даты $currentDate =  
datetime.now()
```

4. Работа с коллекциями — обработка списков, кортежей, словарей.

```
// Пример: сортировка списка $sortedList = list.sort([3, 1,  
2]) // $sortedList = [1, 2, 3]
```

5. Взаимодействие с клиентом — работа с сообщениями.

```
// Пример: получить первое сообщение клиента $firstMessage =  
queue.first().message
```

Советы по работе с функциями

1. Проверяйте доступные модули и функции. Например, модуль `math` содержит математические операции, а модуль `string` — операции со строками.
2. Динамически выбирайте функцию, если это необходимо. Это сделает ваш сценарий более гибким.
3. Передавайте корректные параметры. Ошибки в типах данных могут привести к сбоям.

6.33.4. Переменные

Переменные — это удобный способ сохранять и использовать данные в сценариях ботов. Они работают как именованные «контейнеры», которые бот создает автоматически, когда встречает их впервые. Вначале переменная всегда пустая и имеет значение `nil`.

Типы переменных

Все переменные делятся на три типа, в зависимости от их области видимости:

1. Переменные потока (локальные переменные). Эти переменные работают только в рамках текущего потока — последовательности действий, выполняемых ботом. Они отлично подходят для временного хранения данных, например, промежуточных вычислений. Имена таких переменных начинаются с символа `$`. Пример:

```
$p = 2 * 3.14 * $radius // Переменная $radius доступна только в текущем потоке
```

2. Переменные модуля. Доступны во всех потоках в рамках одного модуля. Это удобно, если нужно обмениваться данными между потоками одного сценария. Имена таких переменных начинаются с символа `#`. Пример:

```
#count = queue.size() // Переменная #count доступна во всех потоках текущего модуля
```

3. Глобальные переменные (переменные окружения). Глобальные переменные видны везде: во всех модулях и потоках. Они содержат информацию о текущем состоянии бота и контексте его работы. Имена таких переменных начинаются с символа `@`. Они доступны только для чтения. Пример:

```
$isChat = @communicationType == "TEXT" // Глобальная переменная @communicationType доступна везде
```

Правила именованя переменных

1. Имена переменных могут содержать:
 - Цифры,
 - Русские и английские буквы,
 - Символ подчеркивания `_`.
2. Нельзя использовать пробелы и специальные символы. Если нужно задать переменную с необычным именем, используйте круглые скобки. Пример:

```
$( '日本' ) = 123 // Создание переменной с именем на японском $x  
= $( '日本' ) // $x теперь равна 123 // Доступ к переменной с  
нестандартным именем
```

Динамические переменные

Иногда нужно создавать имена переменных прямо во время выполнения сценария. Это называется динамическим вычислением имен. Такой подход особенно полезен для сложных логик.

Пример:

```
// Создаем переменные с названиями x1, x2, x3 $x1 = "red" $x2 =  
"blue" $x3 = "green" // Случайное число от 1 до 3 $n = math.rand(1,  
3) // Формируем имя переменной и получаем значение $color = $("x"  
: $n) // Например, $color может стать "red", "blue" или "green"
```

Комплексные примеры

Благодаря динамическим вычислениям переменные могут использовать значения других переменных, что позволяет создавать сложные логические цепочки.

Пример:

```
$x = "y" // $x содержит название переменной #y #y =  
"communicationType" // #y содержит название глобальной переменной  
communicationType $communicationType = @$x // Итоговое значение:  
@communicationType
```

Эффективное использование переменных

- Локальные переменные (`$`) — подходят для временных операций внутри одного потока.
- Модульные переменные (`#`) — используются для обмена данными между потоками в одном модуле.
- Глобальные переменные (`@`) — применяются для получения общей информации о работе бота.

6.34. Функции BPL

6.34.1. Функции общего назначения

Удаление переменной: `del`

Удаляет локальную или модульную переменную.

Сигнатура: `del(varName: string)`

Аргументы: `varName` — название переменной в виде строки. Название переменной может включать префикс области видимости, такой как `$` (для локальных переменных) или `#` (для модульных переменных). Если префикс опущен, переменная считается локальной.

Возвращаемое значение: Нет.

Пример использования:

```
$localVar = 123 #moduleVar = true del("$localVar") // Удаляет локальную переменную $localVar del("localVar") // Тот же эффект, что и del("$localVar") del("#moduleVar") // Удаляет переменную с областью видимости в пределах модуля
```

Изменение временной зоны: `setClientTimezoneOffset`

Изменяет текущее смещение временной зоны пользователя (собеседника бота).

Сигнатура: `setClientTimezoneOffset(offset: int)`

Аргументы: `offset` — смещение временной зоны в минутах. Положительные значения означают смещение на восток, отрицательные — на запад.

Возвращаемое значение: Нет.

Примечание: Вызов этой функции также приведет к изменению значений системных переменных `now`, `today` и `time` в соответствии с новой временной зоной.

Пример использования:

```
setClientTimezoneOffset(-1800) // Устанавливает смещение временной зоны на -30 минут (например, для часового пояса UTC-30)
```

Преобразование в булево значение: `asBool`

Преобразует значение в булев тип.

Сигнатура: `asBool(obj: any) -> bool`

Аргументы: `obj` — любое значение, которое требуется преобразовать.

Возвращаемое значение: Булево значение.

Примечание: В BPL (Basic Programming Language) все значения могут быть преобразованы в булев тип. Например, `nil`, пустые строки или ноль преобразуются в `false`, а все остальные значения — в `true`.

Пример использования:

```
$bool = asBool(nil) // $bool будет равно false $bool = asBool("")
// $bool будет равно false $bool = asBool(0) // $bool будет равно
false $bool = asBool([]) // $bool будет равно true $bool =
asBool(123) // $bool будет равно true
```

Преобразование в строку: `asString`

Преобразует значение в строку.

Сигнатура: `asString(obj: any) -> string`

Аргументы: `*obj` — любое значение, которое требуется преобразовать.

Возвращаемое значение: Строковое представление значения.

Примечание: В BPL все значения могут быть преобразованы в строковый тип.

Пример использования:

```
$str = asString(123) // $str будет равно "123" $str =
asString(1.23) // $str будет равно "1.23" $str = asString(true) //
$str будет равно "true" $str = asString({1: "a b c", 2: 0.5}) //
$str будет равно "{1: "a b c", 2: 0.5}"
```

Преобразование в целое число: `asInt`

Преобразует значение в целое число.

Сигнатура: `asInt(obj: any) -> int`

Аргументы: `obj` — любое значение, которое требуется преобразовать.

Возвращаемое значение: Целочисленное значение. Если преобразование невозможно, возвращается `0`.

Пример использования:

```
$int = asInt(5.67) // $int будет равно 5 $int = asInt("123") //
$int будет равно 123 $int = asInt(true) // $int будет равно 1 $int
= asInt(nil) // $int будет равно 0 $int = asInt("abc") // $int
будет равно 0
```

Преобразование в вещественное число: `asFloat`

Преобразует значение в вещественное число.

Сигнатура: `asFloat(obj: any) -> float`

Аргументы: `obj` — любое значение, которое требуется преобразовать.

Возвращаемое значение: Вещественное число. Если преобразование невозможно, возвращается `0.0`.

Пример использования:

```
$float = asFloat("5.67") // $float будет равно 5.67 $float =  
asFloat(123) // $float будет равно 123.0 $float = asFloat(true) //  
$float будет равно 1.0 $float = asFloat(nil) // $float будет равно  
0.0 $float = asFloat("abc") // $float будет равно 0.0
```

6.34.2. Математические функции

Математические функции — это инструменты для выполнения различных арифметических и числовых операций. Они поддерживают высокую точность вычислений и включают как базовые операции, так и более специализированные. Рассмотрим их подробнее.

Сложение чисел: `math.add`

Выполняет сложение двух чисел. Эквивалент операции `+`.

Сигнатура: `math.add(num1: float, num2: float, precision: int = 12) -> float`

Аргументы:

- `num1` — первое слагаемое.
- `num2` — второе слагаемое.
- (Опционально) `precision` — точность вычислений, количество знаков после запятой. По умолчанию: 12.

Результат: Сумма чисел с заданной точностью.

Пример использования:

```
$a = math.add(1.5, 3.5) // $a равно 5 $a = 1.5 + 3.5 // То же самое  
$a = math.add(1.000006, 2.1, 5) // $a равно 3.10001
```

Разность чисел: `math.sub`

Выполняет вычитание двух чисел. Эквивалент операции `-`.

Сигнатура: `math.sub(num1: float, num2: float, precision: int = 12) -> float`

Аргументы:

- `num1` — уменьшаемое.
- `num2` — вычитаемое.
- (Опционально) `precision` — точность вычислений, количество знаков после запятой. По умолчанию: 12.

Результат: Разность чисел с заданной точностью.

Пример использования:

```
$a = math.sub(1.5, 3.5) // $a равно -2 $a = 1.5 - 3.5 // То же  
самое $a = math.sub(2.100006, 1.1, 5) // $a равно 1.00001
```

Умножение чисел: `math.mul`

Выполняет умножение двух чисел. Эквивалент операции `*`.

Сигнатура: `math.mul(num1: float, num2: float, precision: int = 12) -> float`

Аргументы:

- `num1` — первый множитель.
- `num2` — второй множитель.
- (Опционально) `precision` — точность вычислений, количество знаков после запятой. По умолчанию: 12.

Результат: Произведение чисел с заданной точностью.

Пример использования:

```
$a = math.mul(1.5, 3.5) // $a равно 5.25 $a = 1.5 * 3.5 // То же самое $a = math.mul(1.2345, 1.234, 5) // $a равно 1.52337
```

Деление чисел: `math.div`

Выполняет деление двух чисел. Эквивалент операции `/`.

Сигнатура: `math.div(num1: float, num2: float, precision: int = 12) -> float`

Аргументы:

- `num1` — делимое.
- `num2` — делитель.
- (Опционально) `precision` — точность вычислений, количество знаков после запятой. По умолчанию: 12.

Результат: Частное двух чисел с заданной точностью.

Пример использования:

```
$a = math.div(1.5, 3.5) // $a равно 0.428571428571 $a = 1.5 / 3.5 // То же самое $a = math.div(1, 3, 5) // $a равно 0.333333
```

Целочисленное деление: `math.idiv`

Выполняет целочисленное деление двух чисел. Эквивалент операции `\`.

Сигнатура: `math.idiv(num1: float, num2: float) -> int`

Аргументы:

- `num1` — делимое.
- `num2` — делитель.

Результат: Целая часть частного.

Пример использования:

```
$a = math.idiv(2.5, 0.3) // $a равно 8 $a = 2.5 \ 0.3 // То же самое
```

Остаток от деления: `math.mod`

Вычисляет остаток от деления двух чисел. Эквивалент операции %.

Сигнатура: `math.mod(num1: float, num2: float, precision: int = 12) -> float`

Аргументы:

- `num1` — делимое.
- `num2` — делитель.
- (Опционально) `precision` — точность вычислений, количество знаков после запятой. По умолчанию: 12.

Результат: Остаток от деления.

Пример использования:

```
$a = math.mod(3.5, 1.5) // $a равно 0.5 $a = 3.5 % 1.5 // То же самое $a = math.mod(1/3, 2/7, 5) // $a равно 0.04762
```

Возведение в степень: `math.pow`

Возводит число в указанную степень. Эквивалент операции `**`.

Сигнатура: `math.pow(base: float, power: float, precision: int = 12) -> float`

Аргументы:

- `base` — основание.
- `power` — показатель степени.
- (Опционально) `precision` — точность вычислений. По умолчанию: 12.

Результат: Число, возведенное в степень.

Пример использования:

```
$a = math.pow(1.5, 3.5) // $a равно 4.133513940947 $a = 1.5 ** 3.5 // То же самое $a = math.pow(1.3, 7.1, 5) // $a равно 0.44166
```

Извлечение квадратного корня: `math.sqrt`

Извлекает квадратный корень из числа. Возвращает ошибку, если число отрицательное.

Сигнатура: `math.sqrt(num: float, precision: int = 12) -> float`

Аргументы:

- `num` — число для извлечения корня.
- (Опционально) `precision` — точность вычислений. По умолчанию: 12.

Результат: Квадратный корень числа.

Пример использования:

```
$a = math.sqrt(3.14) // $a равно 1.772004514667 $a = math.sqrt(1.7, 5) // $a равно 0.30384
```

Округление числа: `math.round`

Округляет число до указанного знака после запятой.

Сигнатура: `math.round(num: float, precision: int) -> float`

Аргументы:

- `num` — число для округления.
- `precision` — точность округления.

Результат: Округленное число.

Пример использования:

```
$a = math.round(3.141592653589793, 2) // $a равно 3.14 $a =  
math.round(3.141592653589793, 0) // $a равно 3
```

Генерация случайного числа: `math.rand`

Генерирует псевдослучайное число в заданном диапазоне.

Сигнатура: `math.rand(min: int, max: int) -> int`

Аргументы:

- `min` — минимальное значение.
- `max` — максимальное значение.

Результат: Случайное число в диапазоне.

Пример использования:

```
$r = math.rand(-10, 10)
```

Сравнение чисел: `math.eq` и `math.neq`

Описание:

- `math.eq` проверяет, равны ли числа с заданной точностью.
- `math.neq` проверяет, не равны ли числа с заданной точностью.

Сигнатура:

```
math.eq(a: float, b: float, epsilon: float = 1e-12) -> bool  
math.neq(a: float, b: float, epsilon: float = 1e-12) -> bool
```

Пример использования:

```
$equals = math.eq(0.1, 0.1 + 1e-13) // true $notEquals =  
math.neq(0.1, 0.1 + 1e-11) // false
```

6.34.3. Строковые функции

Функции для работы со строками позволяют анализировать, преобразовывать и манипулировать текстовыми данными.

Определение длины строки: `str.len`

Определяет длину строки в символах.

Сигнатура: `str.len(str: string) -> int`

Аргументы: `str` — строка, длину которой нужно определить.

Результат: Целое число, равное количеству символов в строке.

Пример использования:

```
$str = "Какая-то строка" $len = str.len($str) // $len будет  
содержать 15
```

Преобразование строки в нижний регистр: `str.lower`

Преобразует все символы строки в нижний регистр.

Сигнатура: `str.lower(str: string) -> string`

Аргументы: `str` — строка, которую необходимо преобразовать.

Результат: Строка, все символы которой находятся в нижнем регистре.

Пример использования:

```
$str = "СтРоКа" $lower = str.lower($str) // $lower будет содержать  
"строка"
```

Преобразование строки в верхний регистр: `str.upper`

Преобразует все символы строки в верхний регистр.

Сигнатура: `str.upper(str: string) -> string`

Аргументы: `str` — строка, которую необходимо преобразовать.

Результат: Строка, все символы которой находятся в верхнем регистре.

Пример использования:

```
$str = "СтРоКа" $upper = str.upper($str) // $upper будет содержать  
"СТРОКА"
```

Преобразование первого символа строки в верхний регистр: `str.ucfirst`

Преобразует первый символ строки в верхний регистр.

Сигнатура: `str.ucfirst(str: string) -> string`

Аргументы: `str` — строка, первый символ которой необходимо преобразовать.

Результат: Строка с первым символом в верхнем регистре.

Пример использования:

```
$str = str.ucfirst("строка") // $str будет содержать "Строка"
```

Преобразование первого символа строки в нижний регистр: `str.lcfirst`

Преобразует первый символ строки в нижний регистр.

Сигнатура: `str.lcfirst(str: string) -> string`

Аргументы: `str` — строка, первый символ которой необходимо преобразовать.

Результат: Строка с первым символом в нижнем регистре.

Пример использования:

```
$str = str.lcfirst("Строка") // $str будет содержать "строка"
```

Получение символа строки по индексу: `str.letter`

Возвращает символ строки по указанной позиции.

Сигнатура: `str.letter(str: string, index: int) -> string`

Аргументы:

- `str` — строка, символ которой требуется получить.
- `index` — позиция символа в строке (начиная с 0). Если отрицательная, отсчёт начинается с конца строки.

Результат: Строка, содержащая символ по указанной позиции, или пустая строка, если символа с такой позицией не существует.

Пример использования:

```
$str = "Слово" $firstLetter = str.letter($str, 0) // Первая буква  
$lastLetter = str.letter($str, -1) // Последняя буква
```

Объединение строк: `str.concat`

Объединяет две строки в одну.

Сигнатура: `str.concat(str1: string, str2: string) -> string`

Аргументы:

- `str1` — первая строка.
- `str2` — строка, добавляемая к первой.

Результат: Новая строка, состоящая из первой строки, к которой справа добавлена вторая.

Пример использования:

```
$str1 = "один" $str2 = "два" $str = str.concat($str1, $str2) //  
$str будет содержать "одиндва"
```

6.34.4. Функции хэширования

Функции хэширования используются для вычисления хэш-кодов строки, которые представляют собой уникальные фиксированной длины значения, генерируемые на основе входных данных.

Вычисление хэша строки: `hash.of`

Вычисляет хэш строки с использованием указанного алгоритма.

Сигнатура: `hash.of(text: string, algo: string = "md5", binary: bool = false) -> string`

Аргументы:

- `text` — строка для хэширования.
- `algo` — название алгоритма хэширования (по умолчанию `md5`).
- `binary` — если `true`, возвращает результат в виде двоичных данных. Если `false` (по умолчанию), результат будет в шестнадцатеричной кодировке.

Результат: Возвращает строку с вычисленным хэш-кодом:

- В шестнадцатеричной кодировке, если `binary` установлено в `false`.

- В виде двоичных данных, если `binary` установлено в `true`.
Возвращает пустую строку в случае ошибки (например, при указании недопустимого алгоритма).

Допустимые значения алгоритмов:

`md2`, `md4`, `md5`, `sha1`, `sha224`, `sha256`, `sha384`, `sha512/224`, `sha512/256`, `sha512`,
`sha3-224`, `sha3-256`, `sha3-384`, `sha3-512`, `ripemd128`, `ripemd160`, `ripemd256`,
`ripemd320`,
`whirlpool`, `tiger128,3`, `tiger160,3`, `tiger192,3`, `tiger128,4`, `tiger160,4`,
`tiger192,4`,
`snefru`, `snefru256`, `gost`, `gost-crypto`, `adler32`, `crc32`, `crc32b`, `crc32c`,
`fnv132`, `fnv1a32`, `fnv164`, `fnv1a64`, `joaat`, `murmur3a`, `murmur3c`, `murmur3f`,
`xxh32`, `xxh64`, `xxh3`, `xxh128`, `haval128,3`, `haval160,3`, `haval192,3`, `haval224,3`,
`haval256,3`, `haval128,4`, `haval160,4`, `haval192,4`, `haval224,4`, `haval256,4`,
`haval128,5`, `haval160,5`, `haval192,5`, `haval224,5`, `haval256,5`.

Пример использования:

```
// Хэширование строки с использованием алгоритма по умолчанию (md5)
$hash = hash.of("Наглый коричневый лисёнок прыгает вокруг ленивой
собаки.") // $hash будет содержать
"bff8b4bc8b5c1c1d5b3211dfb21d1e76" // Хэширование строки с
использованием алгоритма ripemd160 $hash = hash.of("Наглый
коричневый лисёнок прыгает вокруг ленивой собаки.", "ripemd160") //
$hash будет содержать "8817ca339f7f902ad3fb456150a1bb9b4cb5dde9" //
Хэширование строки с выводом бинарного результата $hash =
hash.of("Наглый коричневый лисёнок прыгает вокруг ленивой собаки.",
"sha256", true) // $hash будет содержать бинарную строку
(содержащую неотображаемые символы)
```

6.34.5. Кодирование и декодирование

Кодирование в Base64: `codec.base64Encode`

Функция кодирует строку в формат base64, который используется для представления бинарных данных в текстовом виде, что делает их безопасными для передачи по текстовым протоколам.

Сигнатура: `codec.base64Encode(str: string) -> string`

Аргументы: `str` — произвольная строка, которую нужно закодировать в base64.

Результат: Строка, закодированная в формате base64.

Пример использования:

```
$encoded = codec.base64Encode("Привет!") // $encoded будет  
содержать строку "0J/RgNC40LLQtdGCIQ=="
```

Декодирование из Base64: `codec.base64Decode`

Функция декодирует строку, закодированную в формате base64, в исходный текст. Если строка содержит символы, не входящие в стандартный алфавит base64, функция вернет `nil`.

Сигнатура: `codec.base64Decode(str: string) -> string | nil`

Аргументы: `str` — строка, закодированная в base64.

Результат: Исходная строка, если декодирование прошло успешно, или `nil`, если строка содержит недопустимые символы.

Пример использования:

```
$decoded = codec.base64Decode("0J/RgNC40LLQtdGCIQ==") // $decoded  
будет содержать строку "Привет!" $failed =  
codec.base64Decode("Привет!") // $failed будет равен nil
```

URL-кодирование: `codec.urlEncode`

Функция кодирует строку в формат URL. Это полезно, когда необходимо передать строку в URL, так как в нем могут встречаться символы, которые имеют специальное значение (например, пробелы, амперсанды, символы пунктуации). Эти символы заменяются на соответствующие закодированные значения.

Сигнатура: `codec.urlEncode(str: string) -> string`

Аргументы: `str` — строка, которую нужно закодировать для использования в URL.

Результат: Строка, закодированная для безопасного использования в URL. Все символы, не являющиеся буквенно-цифровыми, кроме `-`, `_`, `.` и `~`, будут заменены на знак процента (%) и соответствующие шестнадцатеричные значения.

Пример использования:

```
$encoded = codec.urlEncode("Привет!") // $encoded будет содержать  
строку "%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%82%21"
```

Декодирование URL: `codec.urlDecode`

Функция декодирует строку, закодированную в формате URL. Если строка содержит недопустимый символ после знака процента, функция вернет `nil`.

Сигнатура: `codec.urlDecode(str: string) -> string | nil`

Аргументы: `str` — строка, закодированная для использования в URL.

Результат: Декодированная строка, если она была корректно закодирована. В противном случае возвращается `nil`.

Пример использования:

```
$decoded =
```

```
codec.urlDecode("%D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D1%82%21") //
```

```
$decoded будет содержать строку "Привет!" $failed =
```

```
codec.urlDecode("%%") // $failed будет равен nil
```

6.34.6. Работа с датой и временем

Форматирование даты: `dt.format`

Форматирует дату согласно заданному формату.

Сигнатура: `dt.format(dt string|int|dt.Time, format string) -> string`

Аргументы:

- `dt` — дата в виде строки, целого числа (секунды с 1 января 1970 года) или объекта типа `dt.Time`.
- `format` — строка, определяющая формат вывода даты и времени.

Возвращаемое значение: Строка с датой, отформатированной в соответствии с указанным шаблоном.

Пример использования:

```
$dt = dt.format('2022-12-20 08:34:05.123', 'y.MM.dd h-mm-
```

```
ss.SSSSSS') // "22.12.20 8-34-05.123000" $dt =
```

```
dt.format('16:30:47', '\Hour\s \an\d \minute\s: hh/mm A') // "Hours
```

```
and minutes: 04/30 PM"
```

Допустимые параметры форматирования:

Символ в строке <code>format</code>	Описание	Пример значения
Год		
<code>y</code>	Полное числовое представление года (не менее 4 цифр)	1999, 2012
<code>yy</code>	Два последних числа года, с дополнением нулями	99, 05
Месяц		

M	Порядковый номер месяца без ведущего нуля	1-12
MM	Порядковый номер месяца с ведущим нулем	01-12
День		
d	День месяца без ведущего нуля	1-31
dd	День месяца с ведущим нулем	01-31
Час		
h	Часы в 12-часовом формате без ведущего нуля	1-12
hh	Часы в 12-часовом формате с ведущим нулем	01-12
H	Часы в 24-часовом формате без ведущего нуля	0-23
HH	Часы в 24-часовом формате с ведущим нулем	00-23
a	am или pm в нижнем регистре	am, pm
A	AM или PM в верхнем регистре	AM, PM
Минуты		

<code>m</code>	Минуты без ведущего нуля	0-59
<code>mm</code>	Минуты с ведущим нулем	00-59
Секунды		
<code>s</code>	Секунды без ведущего нуля	0-59
<code>ss</code>	Секунды с ведущим нулем	00-59
Милли/Микро/Нано секунды		
<code>S, SS, SSS, ..., SSSSSSSS</code>	Дробная часть секунды, усеченная до количества цифр	1, 12, 123, 123456
Часовой пояс		
<code>Z</code>	Название часового пояса	UTC, PDT, BST
<code>Z</code>	Смещение часового пояса в часах	+02, -01
<code>ZZ</code>	Смещение часового пояса без двоеточия между часами и минутами	+0200, -0100
<code>ZZ</code>	Смещение часового пояса с двоеточием между часами и минутами	+02:00, -01:00

<code>zzz</code>	Смещение часового пояса без двоеточия между часами, минутами и секундами	+020000, -010000
<code>ZZZ</code>	Смещение часового пояса с двоеточием между часами, минутами и секундами	+02:00:00, -01:00:00

6.34.7. Управление очередью сообщений пользователя

Получение размера очереди сообщений

Возвращает размер очереди входящих сообщений, то есть количество сообщений, отправленных пользователем.

Сигнатура: `input.size() -> int`

Аргументы: Отсутствуют.

Результат: Число входных сообщений, отправленных собеседником боту.

Пример использования:

```
$messageCount = input.size() // $messageCount равно числу
сообщений, отправленных ботом
```

Получение последнего сообщения

Возвращает последнее входящее сообщение, либо `nil`, если очередь сообщений пуста.

Сигнатура: `input.last() -> input.Message | nil`

Аргументы: Отсутствуют.

Результат: Объект `input.Message`, представляющий последнее сообщение, или `nil`, если сообщений нет.

Пример использования:

```
$lastMessage = input.last() // $lastMessage содержит последнее
сообщение, отправленное ботом
```

Получение первого сообщения

Возвращает первое сообщение, отправленное собеседником, либо `nil`, если очередь сообщений пуста.

Сигнатура: `input.first() -> input.Message | nil`

Аргументы: Отсутствуют.

Результат: Объект `input.Message`, представляющий первое сообщение, или `nil`, если сообщений нет.

Пример использования:

```
$firstMessage = input.first() // $firstMessage содержит первое сообщение, отправленное ботом
```

Получение сообщения по порядковому номеру

Возвращает сообщение по его порядковому номеру, начиная с 0. Можно использовать отрицательные числа для отсчета с конца очереди.

Сигнатура: `input.nth(index: int) -> input.Message | nil`

Аргументы: `index` — порядковый номер сообщения. Если число отрицательное, отсчет ведется с конца очереди.

Результат: Объект `input.Message` или `nil`, если указанного сообщения не существует.

Пример использования:

```
$message = input.nth(0) // $message содержит первое сообщение, отправленное ботом
$message = input.nth(4) // $message содержит пятое сообщение, отправленное ботом
$message = input.nth(-1) // $message содержит последнее сообщение, отправленное ботом
$message = input.nth(-3) // $message содержит третье сообщение с конца
```

Получение первого необработанного сообщения

Возвращает первое необработанное сообщение. Если таких сообщений нет или очередь пуста, возвращает `nil`.

Сигнатура: `input.firstUnprocessed() -> input.Message | nil`

Аргументы: Отсутствуют.

Результат: Объект `input.Message` с первым необработанным сообщением или `nil`, если таких сообщений нет.

Пример использования:

```
$message = input.firstUnprocessed() // $message содержит первое необработанное сообщение
```

Получение последнего необработанного сообщения

Возвращает последнее необработанное сообщение. Если таких сообщений нет или очередь пуста, возвращает `nil`.

Сигнатура: `input.lastUnprocessed() -> input.Message | nil`

Аргументы: Отсутствуют.

Результат: Объект `input.Message` с последним необработанным сообщением или `nil`, если таких сообщений нет.

Пример использования:

```
$message = input.lastUnprocessed() // $message содержит последнее  
необработанное сообщение
```

6.34.8. Обработка сообщений бота

Ожидание сообщения от собеседника бота: `trigger.input()`

Ожидает сообщение от собеседника бота, при этом приостанавливает выполнение текущего потока, пока не получит входное сообщение.

Аргументы: Нет.

Пример использования:

```
trigger.input(); // Поток приостанавливается до получения сообщения  
от собеседника бота
```

Ожидание ошибки: `trigger.error()`

Возвращает текст ошибки, если одна из подписанных на данный триггер потоков завершилась с ошибкой.

Аргументы: Нет.

Пример использования:

```
$error = trigger.error(); // Сохраняет текст ошибки, если она  
произошла в одном из потоков
```

Установка обработчика ошибок: `trigger.setErrorHandler()`

Устанавливает обработчик ошибок для текущего потока. Когда происходит ошибка, управление передается в указанный поток-обработчик.

Аргументы:

- `startNodeId` — идентификатор узла, с которого начнется выполнение потока.
- `handlerThreadId` (необязательный) — идентификатор потока-обработчика ошибок. Если не указан, будет сгенерирован новый идентификатор.

Пример использования:

```
$handlerThreadId = trigger.setErrorHandler("abbbc08d-41a1-4d09-  
9111-c515ea79634d"); // Устанавливаем обработчик ошибок, начиная с  
узла "abbbc08d-41a1-4d09-9111-c515ea79634d"
```

Удаление обработчика ошибок: `trigger.removeErrorHandler()`

Отключает текущий поток от потока-обработчика ошибок, прекращая его обработку ошибок.

Аргументы: `handlerThreadId` — идентификатор потока-обработчика ошибок, который необходимо отключить.

Пример использования:

```
trigger.removeErrorHandler("abbbc08d-41a1-4d09-9111-c515ea79634d");  
// Удаляет обработчик ошибок с указанным идентификатором
```

6.34.9. Взаимодействие с файлами

Загрузка файла: `file.download`

Загружает файл по указанной интернет-ссылке и возвращает идентификатор загруженного файла.

Сигнатура: `file.download(url: string) -> string`

Аргументы: `url` — строка, содержащая URL-адрес файла, который необходимо загрузить.

Возвращаемое значение: Строка, представляющая идентификатор файла, загруженного в систему TWIN.

Примечание: Убедитесь, что указанный URL корректен и доступен для загрузки. Если загрузка не удалась (например, из-за недоступности ресурса), функция может вернуть ошибку или пустой идентификатор.

Пример использования:

```
$fileId = file.download("http://some.file.url") // $fileId будет  
содержать идентификатор загруженного файла
```

В приведенном примере файл скачивается с указанного URL, а его идентификатор сохраняется в переменной `$fileId`. Этот идентификатор может быть использован для дальнейшей обработки файла в системе.

Получение информации о загруженном файле: `file.info`

Функция `file.info` используется для получения данных о файле, который был загружен ранее. Она возвращает подробную информацию о файле в виде ассоциативного массива (`map`), включая идентификатор, имя, тип содержимого и другие метаданные.

Сигнатура: `file.info(fileId: string) -> Map`

Назначение: Предоставление информации о загруженном в систему файле.

Аргументы: `fileId` — уникальный идентификатор файла, информация о котором требуется.

Возвращаемое значение: Ассоциативный массив (`map`) со следующей структурой:

```
{ "id": "ec8822eb-4055-4190-9c32-9f482aeadecf", // Уникальный  
идентификатор файла "createdAt": "2024-12-16T09:07:15+00:00", //  
Время создания файла в ISO 8601 "contentType": "image/jpeg", // Тип
```

```
содержимого файла "name": "images.jpg", // Полное имя файла
"baseName": "images", // Имя файла без расширения "extension":
"jpg", // Оригинальное расширение файла "suggestedExtension":
"jpg", // Рекомендуемое расширение (определено на основе типа
содержимого) "size": 11638 // Размер файла в байтах }
```

Пример использования:

1. Загрузка файла. Сначала файл загружается в систему, например, с указанного URL:

```
$fileId = file.download("https://example.com/somefile.jpg");
```

2. Получение информации о файле. После загрузки информация о файле может быть получена по его идентификатору:

```
$info = file.info($fileId);
```

3. Результат работы функции, который будет содержать метаданные загруженного файла:

```
{ "id": "ec8822eb-4055-4190-9c32-9f482aeadecf", "createdAt":
"2024-12-16T09:07:15+00:00", "contentType": "image/jpeg",
"name": "images.jpg", "baseName": "images", "extension":
"jpg", "suggestedExtension": "jpg", "size": 11638 }
```

6.34.10. Операции с фактами

Функции для работы с фактами обеспечивают управление базой фактов, позволяя сохранять, извлекать, удалять и очищать данные.

Сохранение факта: `fact.save`

Сохраняет факт в базе данных.

Сигнатура: `fact.save(context: string, factName: string, factValue: mixed, botId: string = nil, clientId: string = nil)`

Аргументы:

- `context` — строка, задающая контекст, в рамках которого существует факт.
- `factName` — строка, задающая название факта.
- `factValue` — значение, представляющее содержимое факта.
- (Опционально) `botId` — идентификатор бота.
- (Опционально) `clientId` — идентификатор клиента.

Возвращаемое значение: Отсутствует.

Пример использования:

```
fact.save("место", "город", "Екатеринбург") // Факт доступен всем ботам компании
fact.save("место", "город", "Екатеринбург", nil, @clientId) // Факт привязан к клиенту
fact.save("место", "город", "Екатеринбург", @botId) // Факт привязан к боту
fact.save("место", "город", "Екатеринбург", @botId, @clientId) // Факт привязан к боту и клиенту
```

Загрузка факта: `fact.load`

Извлекает факт из базы данных.

Сигнатура: `fact.load(context: string, factName: string, botId: string = nil, clientId: string = nil) -> mixed`

Аргументы:

- `context` — строка, задающая контекст.
- `factName` — строка, задающая название факта.
- (Опционально) `botId` — идентификатор бота.
- (Опционально) `clientId` — идентификатор клиента.

Возвращаемое значение:

Содержимое факта.

Пример использования:

```
fact.save("место", "город", "Екатеринбург", @botId, @clientId) // Сохраняем факт
$city = fact.load("место", "город", @botId, @clientId) // Загружаем факт. $city содержит "Екатеринбург"
```

Удаление факта: `fact.delete`

Удаляет факт из базы данных.

Сигнатура: `fact.delete(context: string, factName: string, botId: string = nil, clientId: string = nil)`

Аргументы:

- `context` — строка, задающая контекст.
- `factName` — строка, задающая название факта.
- (Опционально) `botId` — идентификатор бота.
- (Опционально) `clientId` — идентификатор клиента.

Возвращаемое значение: Отсутствует.

Пример использования:

```
fact.save("место", "город", "Екатеринбург", @botId, @clientId) // Сохраняет факт с привязкой к боту и клиенту
$city =
```

```
fact.load("место", "город", @botId, @clientId) // Загружаем факт в
переменную. $city содержит "Екатеринбург" fact.delete("место",
"город", @botId, @clientId) // Удаляем факт $city =
fact.load("место", "город", @botId, @clientId) // Пытаемся
загрузить удаленный факт. Теперь $city содержит nil.
```

Очистка базы фактов: `fact.clean`

Функция `fact.clean` удаляет все факты, соответствующие переданным аргументам. Важно учитывать, что если переданы `botIds` или `clientIds`, то удаляются все факты, относящиеся к указанным ботам или клиентам, вне зависимости от значений `contexts`, `factNames` и `factValues`. Функция не поддерживает удаление на основе условий.

Функция `fact.clean` не позволяет задавать условия удаления. Она удаляет все факты, соответствующие переданным аргументам, без учета их пересечений. Перед использованием убедитесь, что удаляемые данные соответствуют вашим ожиданиям. Сигнатура: `fact.clean(contexts: string|List, factNames: string|List = nil, factValues: any = nil, botIds: string|List = nil, clientIds: string|List = nil)`

Аргументы:

- `contexts` — строка или список строк, определяющий контексты.
- (Опционально) `factNames` — строка или список строк, задающий названия фактов.
- (Опционально) `factValues` — значение или список значений фактов.
- (Опционально) `botIds` — идентификатор или список идентификаторов ботов. При передаче удаляются все факты, относящиеся к указанным ботам, вне зависимости от других аргументов.
- (Опционально) `clientIds` — идентификатор или список идентификаторов клиентов. При передаче удаляются все факты, относящиеся к указанным клиентам, вне зависимости от других аргументов.

Возвращаемое значение: Отсутствует.

Примеры использования:

1. Удаление фактов с определенным контекстом (будут удалены для всех ботов и клиентов):

```
fact.clean("Контекст", ["fact1", "fact2"], [123, 456])
```

2. Удаление всех фактов, связанных с конкретным ботом (независимо от контекста и значений фактов):

```
fact.clean(nil, nil, nil, "fa5d268c-bcc9-4734-a10f-3dfd357764ac")
```

3. Удаление всех фактов, относящихся к конкретному клиенту

```
fact.clean(nil, nil, nil, nil, @clientId)
```

4. Удаление всех фактов, относящихся к конкретному боту и клиенту (каждому отдельно, а не их пересечению):

```
fact.clean(nil, nil, nil, "fa5d268c-bcc9-4734-a10f-3dfd357764ac", @clientId)
```

5. Этот вызов удалит все факты, относящиеся к переданному боту и переданному клиенту, даже если они не связаны между собой.

6.34.11. Функции таймера

Функции таймера управляют обратным отсчетом для выполнения переходов или выполнения задач.

Запуск таймера: `timer.start`

Запускает таймер обратного отсчета.

Сигнатура: `timer.start(time: int, nodeId: string) -> string`

Аргументы:

- `time` — время в секундах.
- `nodeId` — идентификатор блока для перехода после завершения отсчета.

Возвращаемое значение: Идентификатор таймера.

Пример использования:

```
$timerId = timer.start(60, "760b9732-4bfb-4846-a348-faae5138fcb2")  
// Таймер на 60 секунд
```

Остановка таймера: `timer.stop`

Останавливает (удаляет) таймер обратного отсчета.

Сигнатура: `timer.stop(timerId: string)`

Аргументы: `timerId` — идентификатор таймера.

Возвращаемое значение: Отсутствует.

Пример использования:

```
$timerId = timer.start(60, "760b9732-4bfb-4846-a348-faae5138fcb2")  
// Запуск таймера timer.stop($timerId) // Остановка таймера
```

Остановка всех таймеров: `timer.stopAll`

Останавливает (удаляет) все активные таймеры.

Сигнатура: `timer.stopAll()`

Аргументы: Отсутствуют.

Возвращаемое значение: Отсутствует.

Пример использования:

```
timer.start(60, "760b9732-4bfb-4846-a348-faae5138fcb2") // Таймер 1
timer.start(120, "760b9732-4bfb-4846-a348-faae5138fcb2") // Таймер
2 timer.stopAll() // Остановка всех таймеров
```

6.34.12. Работа с текстом на естественном языке NLP

NLP-функции позволяют обрабатывать сообщения, извлекать из них намерения и сущности, а также управлять обработкой текста.

Парсинг текста: `nlp.parse`

Парсит текст на естественном языке, выделяя намерения и сущности.

Сигнатура: `nlp.parse(message: string|UserMessage) -> Sentence`

Аргументы: `message` — текст сообщения или объект `UserMessage`.

Возвращаемое значение: Объект `Sentence`, содержащий информацию о намерениях и сущностях.

Примечание: Функция извлекает только числовые и временные сущности. Для обработки других типов используйте `nlu.parse`.

Пример использования:

```
$sentence = nlp.parse(queue.first()) // Парсим первое сообщение
пользователя
```

Объединение и парсинг текста: `nlp.join`

Объединяет два сообщения в одно и затем парсит их.

Сигнатура: `nlp.join(message1: string|UserMessage, message2: string|UserMessage) -> Sentence`

Аргументы:

- `message1` — первое сообщение.
- `message2` — второе сообщение.

Возвращаемое значение: Объект `Sentence`, содержащий информацию о намерениях и сущностях объединённого текста.

Пример использования:

```
$sentence = nlp.join(queue.lastNth(2), queue.lastNth(1)) //
Объединяем два последних сообщения и парсим
```

Установка восприятия текста: `nlp.setPerception`

Устанавливает сообщение пользователя для дальнейшей обработки другими узлами схемы.

Сигнатура: `nlp.setPerception(sentence: Sentence)`

Аргументы: `sentence` — объект `Sentence`, содержащий информацию о намерениях и сущностях.

Возвращаемое значение: Отсутствует.

Пример использования:

```
$sentence = nlp.join(queue.lastNth(2), queue.lastNth(1)) //
```

```
Объединяем два сообщения nlp.setPerception($sentence) //
```

```
Устанавливаем восприятие для остальных узлов
```

6.34.13. Понимание естественного языка

Разбор текста: `nlu.parse`

Функция `nlu.parse` используется для анализа текста и выявления намерений (что хочет сказать пользователь) и сущностей (конкретных объектов в тексте, таких как даты, локации, имена). Функция обрабатывает текст и возвращает объект, содержащий результаты анализа.

Сигнатура: `nlu.parse(text: string, agentId: string, modelId: string = "", confidence: float = 1, timezone: string = "UTC", time: int|string|dt.Time = nil) -> nlu.Recognition`

Аргументы:

- `text` — текст сообщения, который необходимо проанализировать.
- `agentId` — уникальный идентификатор NLU-агента, который будет использоваться для распознавания. Это обязательный параметр.
- `modelId` — версия агента, которую нужно использовать. Этот параметр является необязательным. Если не указан, будет использована последняя версия агента.
- `confidence` — минимальный порог доверия для распознавания намерений. Значение должно быть в пределах от 0 до 1. Если ни одно намерение не достигает этого порога, будет выбрано намерение с максимальной достоверностью.
- `timezone` — часовой пояс, который используется для корректного распознавания временных сущностей. По умолчанию используется «UTC».
- `time` — текущее время, которое может быть использовано для корректировки распознавания временных сущностей. Если параметр не задан, будет использовано текущее системное время.

Возвращаемое значение: Функция возвращает объект типа `nlu.Recognition`, который содержит результат анализа текста, включая распознанное намерение и сущности.

Пример использования:

```
$recognition = nlu.parse("Привет, Вася", "1ee8b80f-b5db-47b8-9786-70c2992da713",  
" ", 0.6)
```

В этом примере происходит анализ текста "Привет, Вася", используя агент с идентификатором 1ee8b80f-b5db-47b8-9786-70c2992da713. Результатом работы будет объект, содержащий распознанное намерение и извлеченные сущности.

Примечания:

- Параметры `timezone` и `time` особенно важны для правильной обработки временных сущностей (например, даты и времени). Убедитесь, что они указаны корректно, если это необходимо для вашего случая.
- Параметр `confidence` позволяет контролировать точность распознавания намерений. Если значения доверия для всех намерений ниже указанного порога, будет использовано намерение по умолчанию.

Рекомендации на основе текста: `nlu.suggest`

Функция `nlu.suggest` помогает сопоставить текст пользователя с заданными рекомендациями, основываясь на намерениях и сущностях.

Сигнатура: `nlu.suggest(text: string, agentId: string, timezoneOffset: int, version: int, confidence: float, suggestions: array, maxSuggestionCount: int) -> array`

Аргументы:

- `text` — текст для анализа.
- `agentId` — идентификатор агента, выполняющего анализ.
- `timezoneOffset` — смещение временной зоны для корректного выявления временных сущностей.
- `version` — версия агента (1 — для старых сценариев, 11 — для новых).
- `confidence` — минимальный порог доверия для распознавания намерений.
- `suggestions` — список рекомендаций с указанием намерений и сущностей, например:

```
[{"intent": "buy", "entities": ["apple", "orange"]}, {"intent": "sell", "entities": ["juice"]}]
```
- `maxSuggestionCount` — максимальное количество возвращаемых рекомендаций (0 или отрицательное значение отключают ограничение).

Возвращаемое значение: Список рекомендаций, подходящих по указанным намерениям и сущностям. К каждой рекомендации добавляется параметр `confidence`, указывающий степень уверенности системы.

Примеры использования:

1. Пользователь хочет купить:

```
$suggestions = [ { "message": "Что вы хотите купить?", "intent": "buy" }, { "message":  
"Что вы хотите продать?", "intent": "sell", "entities": ["juice"] } ]; $result =  
nlu.suggest("Я хочу купить яблоки", $agentId, 0, 1, 0.89, $suggestions, -1);
```

Результат:

```
[ { "message": "Что вы хотите купить?", "intent": "buy", "confidence": 0.97 } ]
```

2. Пользователь хочет продать:

```
$result = nlu.suggest("Я хочу продать сок", $agentId, 0, 1, 0.89, $suggestions, -1);
```

Результат:

```
[ { "message": "Что вы хотите продать?", "intent": "sell", "entities": ["juice"],  
"confidence": 0.95 } ]
```

Примечания:

- Укажите параметры `timezone` и `time`, если точность работы с временными сущностями критична.
- Используйте параметр `confidence` для настройки чувствительности системы к различным намерениям.
- Используйте параметр `maxSuggestionCount` для настройки ограничения количества рекомендаций.

6.34.14. Функции для работы с HTTP

Основная функция: `http.sendRequest`

Отправляет HTTP-запрос к указанному URL.

Сигнатура: `http.sendRequest(url string, method string, body any = nil, headers Map = nil) Response`

Аргументы:

- `url` — строка, содержащая URL-адрес.
- `method` — HTTP-метод. Поддерживаемые значения: `GET`, `POST`, `PUT`, `DELETE`, `PATCH`, `HEAD`, `OPTIONS`.
- `body` — тело запроса. Автоматически преобразуется в JSON или XML, если указан соответствующий тип содержимого.
- `headers` — карта заголовков запроса. По умолчанию устанавливается `Content-Type: application/json`.

Возвращаемое значение: Объект ответа, содержащий:

- `statusCode` — HTTP-код состояния.
- `body` — тело ответа. Автоматически преобразуется из JSON или XML в структуру данных.
- `headers` — заголовки ответа.

- `error` — ошибки (если есть).

Пример JSON-запроса:

Отправка GET-запроса с заголовками:

```
$response = http.sendRequest( "https://example.com/api/v1/users", "GET", {"limit": 10, "offset": 0}, {"Authorization": "Bearer myAccessToken"} ) $statusCode = $response.statusCode // HTTP-статус: 200 $body = $response.body // Данные ответа в формате JSON (ассоциативный массив) $headers = $response.headers // Заголовки ответа
```

Отправка POST-запроса с телом в формате JSON:

```
$requestBody = { "username": "johndoe", "email": "johndoe@example.com" } $response = http.sendRequest( "https://example.com/api/v1/register", "POST", $requestBody, {"Content-Type": "application/json"} ) $body = $response.body // Ответ в виде ассоциативного массива
```

Пример XML-запроса:

Отправка XML-запроса с преобразованием структуры в строку:

```
$requestBody = [ { "tag": "request", "value": [ { "tag": "user", "value": [ { "tag": "name", "value": "John Doe" }, { "tag": "email", "value": "john.doe@example.com" } ] } ] } ] $response = http.sendRequest( "https://example.com/api/v1/xml", "POST", $requestBody, {"Content-Type": "text/xml"} ) $body = $response.body // Ответ в формате XML преобразуется в структуру
```

Создание запросов: `http.request`

Создает объект HTTP-запроса, который можно настраивать перед отправкой.

Сигнатура: `http.request(url string = "", method string = "POST", body any = nil) Request`

Аргументы:

- `url` — строка URL.
- `method` — HTTP-метод.
- `body` — тело запроса.

Возвращаемое значение: Объект запроса с методами для дальнейшей настройки:

- `headers(map)` — установка заголовков.
- `timeout(seconds)` — установка тайм-аута.
- `send()` — отправка запроса.

Пример:

Создание и отправка POST-запроса:

```
$response = http.request("https://api.example.com/data", "POST", {"key": "value"})  
.headers({"Content-Type": "application/json", "Authorization": "Bearer token"})  
.timeout(60) .send() $statusCode = $response.statusCode // HTTP-статус $body =  
$response.body // Тело ответа
```

Создание GET-запроса с минимальными настройками:

```
$response = http.request("https://api.example.com/status", "GET").send()
```

Примечания:

1. Автоматическая обработка JSON и XML:
 - Если содержимое запроса или ответа имеет тип `application/json`, тело автоматически преобразуется в соответствующие структуры данных.
 - Аналогично, содержимое типа `text/xml` обрабатывается как XML.
2. Проверьте поле `$response.error`, чтобы убедиться, что запрос выполнен успешно.

6.34.15. Системные функции

Пауза в исполнении: `sys.sleep`

Останавливает работу бота на указанное количество микросекунд. Если количество микросекунд превышает 1 минуту, пауза будет ограничена до 1 минуты.

Сигнатура: `sys.sleep(microseconds: int)`

Аргументы: `microseconds` — количество микросекунд для паузы. Если значение превышает 60 секунд (1 минута), пауза будет ограничена 1 минутой.

Возвращаемое значение: Нет.

Пример использования:

```
sys.sleep(3_000_000) // Пауза в 3 секунды
```

6.34.16. Функции GPT

Запрос к нейросети: `gpt.ask`

Отправляет сообщение в нейросеть ChatGPT и возвращает её ответ.

Сигнатура: `gpt.ask(model: string, text: string, temperature: float = 0.7, useContext: bool = false, maxTokens: int = 0, timeout: int = 0) -> string`

Аргументы:

- `model` — название конкретной модели нейросети.
- `text` — запрос к нейросети на русском языке.

- `temperature` — число от 0 до 1, обозначающее степень достоверности и вариативности ответов нейросети. (0 — максимальная достоверность и минимальная вариативность, 1 — минимальная достоверность и максимальная вариативность).
- `useContext` — определяет, следует ли использовать предыдущий контекст разговора или нет.
- `maxTokens` — если больше нуля, то определяет максимальное число токенов в ответе нейросети. Если меньше или равно нулю, не используется.
- `timeout` — если больше нуля, то ограничивает время выполнения запроса (в секундах). Если меньше или равно, не используется.

Возвращаемое значение: Ответ нейросети в виде строки.

Примечание: Если параметр `useContext` равен `true`, то текущее сообщение будет добавлено в переменную списка `"context.{MODEL}"`. Все предыдущие сообщения из этого списка будут переданы в качестве контекста запроса. Если параметр `useContext` равен `false`, текущее сообщение не добавляется в список.

Пример использования:

```
$answer = gpt.ask("gpt-4-1106-preview", "Есть ли жизнь на Марсе?", 0.5, true, 500, 10); // В $answer будет ответ нейросети на заданный вопрос.
```

Создание разговорной сессии: `gpt.createThread`

Создает разговорную сессию с GPT ассистентом.

Сигнатура: `gpt.createThread() -> string`

Аргументы: Отсутствуют.

Возвращаемое значение: Идентификатор чат сессии с ассистентом.

Пример использования:

```
$threadId = gpt.createThread(); // В $threadId будет идентификатор новой чат сессии.
```

Удаление разговорной сессии: `gpt.deleteThread`

Удаляет разговорную сессию с GPT ассистентом.

Сигнатура: `gpt.deleteThread(threadId: string)`

Аргументы: `threadId` — идентификатор чат сессии.

Возвращаемое значение: Отсутствует.

Пример использования:

```
gpt.deleteThread($threadId); // Удаляет чат сессию, идентификатор которой сохранён в $threadId.
```

Отправка сообщения в ассистента: `gpt.assist`

Отправляет сообщение в ассистента ChatGPT и возвращает его ответ.

Сигнатура: `gpt.assist(assistantId: string, threadId: string, messages: string|Collection, model: string = '', instructions: string = '', additionalInstructions: string = '', temperature: float = 0.7, maxTokens: int = 0, timeout: int = 0) -> string`

Аргументы:

- `assistantId` — идентификатор ассистента.
- `threadId` — идентификатор чат сессии с ассистентом, полученный с помощью функции `gpt.createThread()`.
- `messages` — запрос к нейросети, может быть как строкой, так и списком из нескольких сообщений.
- `model` — название модели, которая будет применяться вместо той модели, которая была указана при создании ассистента.
- `instructions` — указания для ассистента, которые будут использоваться вместо тех, что были указаны при его создании.
- `additionalInstructions` — дополнительная инструкция, которая будет добавлена в конец инструкции ассистента.
- `temperature` — число от 0 до 1, обозначающее степень достоверности и вариативности ответов нейросети. (0 — максимальная достоверность и минимальная вариативность, 1 — минимальная достоверность и максимальная вариативность).
- `maxTokens` — если больше нуля, то определяет максимальное число токенов в ответе нейросети. Если меньше или равно нулю, не используется.
- `timeout` — если больше нуля, то ограничивает время выполнения запроса (в секундах). Если меньше или равно, не используется.

Возвращаемое значение: Ответ нейросети в виде строки.

Пример использования:

```
$answer = gpt.assist($assistantId, $threadId, "Есть ли жизнь на Марсе?", "gpt-3.5-turbo-1106", "", "", 0.5, 500, 10); // В $answer будет ответ нейросети на заданный вопрос.
```

6.34.17. Функции RekaAI

Запрос к нейросети: `reka.ask`

Отправляет сообщение в нейросеть Reka и возвращает её ответ.

Сигнатура: `reka.ask(model: string, text: string, temperature: float = 0.7, maxTokens: int = 0, timeout: int = 0) -> string`

Аргументы:

- `model` — название модели (например, «reka-core» для Reka Core или «g» для Reka Flash).
- `text` — запрос к нейросети на русском языке.

- `temperature` — число от 0 до 1, обозначающее степень достоверности и вариативности ответов нейросети. Чем ближе к 0, тем более достоверные, но менее вариативные ответы. Чем ближе к 1, тем более вариативные, но менее достоверные ответы.
- `maxTokens` — максимальное число токенов в ответе нейросети. Если меньше или равно нулю, не используется.
- `timeout` — время выполнения запроса в секундах. Если меньше или равно нулю, не используется.

Возвращаемое значение: Ответ нейросети в виде строки.

Возможные значения для параметра `model`:

- `reka-core` — для Reka Core.
- `r` — для Reka Flash.

Пример использования:

```
$answer = reka.ask("reka-core", "Есть ли жизнь на Марсе?", 0.5, 500, 15); // В $answer будет содержаться ответ нейросети на заданный вопрос.
```

6.34.18. Функции GigaChat

Запрос к нейросети: `gigachat.ask`

Отправляет сообщение в нейросеть GigaChat и возвращает её ответ.

Сигнатура: `gigachat.ask(model: string, text: string, temperature: float = 0.7, maxTokens: int = 0, timeout: int = 0) -> string`

Аргументы:

- `model` — название модели (например, «GigaChat» для GigaChat Lite или «GigaChat-Pro» для GigaChat Pro).
- `text` — запрос к нейросети на русском языке.
- `temperature` — число от 0 до 1, обозначающее степень достоверности и вариативности ответов нейросети. Чем ближе к 0, тем более достоверные, но менее вариативные ответы. Чем ближе к 1, тем более вариативные, но менее достоверные ответы.
- `maxTokens` — максимальное число токенов в ответе нейросети. Если меньше или равно нулю, не используется.
- `timeout` — время выполнения запроса в секундах. Если меньше или равно нулю, не используется.

Возвращаемое значение: Ответ нейросети в виде строки.

Возможные значения для параметра `model`:

- `GigaChat` — для использования GigaChat Lite.
- `GigaChat-Pro` — для использования GigaChat Pro.

Пример использования:

```
$answer = gigachat.ask("GigaChat", "Есть ли жизнь на Марсе?", 0.5, 500, 15); // В
```

`$answer` будет содержать ответ нейросети на заданный вопрос.

6.34.19. Функции YandexGPT

Запрос к нейросети: `ygpt.ask`

Отправляет сообщение в нейросеть YandexGPT и возвращает её ответ.

Сигнатура: `ygpt.ask(model: string, text: string, temperature: float = 0.7, maxTokens: int = 0, timeout: int = 0) -> string`

Аргументы:

- `model` — название модели:
- `yandexgpt` — для YandexGPT Pro.
- `yandexgpt-lite` — для YandexGPT Lite.
- `summarization` — для формирования краткого пересказа текста.
- `ds://<идентификатор_дообученной_модели>` — для моделей, дообученных в Yandex DataSphere.
- Возможность указания версии модели через «/latest», «/rc», «/deprecated».
- `text` — запрос к нейросети на русском языке.
- `temperature` — число от 0 до 1, обозначающее степень достоверности и вариативности ответов нейросети. Чем ближе к 0, тем более достоверные, но менее вариативные ответы. Чем ближе к 1, тем более вариативные, но менее достоверные ответы.
- `maxTokens` — максимальное число токенов в ответе нейросети. Если меньше или равно нулю, не используется.
- `timeout` — время выполнения запроса в секундах. Если меньше или равно нулю, не используется.

Возвращаемое значение: Ответ нейросети в виде строки.

Примечание: Для каждой модели можно указать версию, используя «/latest», «/rc», или «/deprecated». Соответствие версии и модели можно найти в официальной документации Yandex.

Пример использования:

```
$answer = ygpt.ask("yandexgpt/latest", "Есть ли жизнь на Марсе?", 0.5, 500, 15); # В $answer
```

будет содержаться ответ нейросети на заданный вопрос.

6.34.20. Функции YCLIENTS

Запись клиента на услугу: `yclients.createRecord`

Записывает клиента на услугу.

Сигнатура: `yclients.createRecord(salonId: int, params: Map) -> ?int`

Аргументы:

- `salonId` — идентификатор филиала.

- `params` — параметры записи:
- `staffId` — идентификатор сотрудника (обязательный).
- `services` — список услуг (обязательный). Каждый элемент — ассоциативный массив:
- `id` — идентификатор услуги.
- `client` — информация о клиенте (обязательная):
- `phone` — номер телефона клиента (обязательный).
- `name` — имя клиента (обязательное для нового клиента).
- `email` — email клиента.
- `datetime` — дата и время записи (обязательная).
- `seanceLength` — длительность сеанса в секундах (обязательная).
- `saveIfBusy` — сохранять запись, если время занято (по умолчанию `false`).
- `sendSms` — отправлять ли СМС с деталями записи клиенту (по умолчанию `false`).
- `comment` — комментарий к записи.
- `smsRemainHours` — за сколько часов до визита отправить СМС напоминание (по умолчанию 1).
- `emailRemainHours` — за сколько часов до визита отправить email напоминание (по умолчанию 12).
- `attendance` — статус записи (0 — ожидание, 1 — услуги оказаны, 2 — подтверждена, -1 — не пришел).
- `customFields` — дополнительные поля, заполненные согласно настройкам филиала.
- `recordLabels` — список идентификаторов категорий записи.
- `customColor` — цвет записи (по умолчанию `null`).
- `apiId` — идентификатор внешней системы (по умолчанию `null`).

Возвращаемое значение: `int` — идентификатор записи в случае успеха или `nil` в случае ошибки.

Пример использования:

```
$recordId = yclients.createRecord(25344, { "staffId": 2303331, "services": [{"id": 11428840}], "client": { "phone": "79876543210", "name": "Семён", "email": "semen@gmail.com" }, "datetime": "2023-06-07 15:00", "seanceLength": 3600, "saveIfBusy": false, "sendSms": false, "comment": "Комментарий к записи", "smsRemainHours": 1, "emailRemainHours": 12, "attendance": 2, "customFields": {"priority": "high"}, "recordLabels": ["67345", "78549"], "customColor": null, "apiId": "7894" });
```

Детали записи: `yclients.recordDetails`

Получает данные о записи.

Сигнатура: `yclients.recordDetails(salonId: int, recordId: int) ->`

`?Map`

Аргументы:

- `salonId` — идентификатор филиала.
- `recordId` — идентификатор записи.

Возвращаемое значение: Ассоциативный массив с деталями записи, если запись существует, или `nil` в случае ошибки.

Пример использования

```
$record = yclients.recordDetails(25344, 52157914); // $record будет содержать: // { // "id": 52157914, // "client": { // "id": 167359987, // "name": "Семён", // "surname": "", // "phone": "+79876543210", // "card": "", // "email": "semen@gmail.com" // }, // "staff": { // "id": 2303331, // "name": "Анисимова Полина", // "specialization": "специалист", // "position": { // "id": 231647, // "title": "Парикмахер" // } // }, // "services": [ // { // "id": 11440288, // "title": "Стрижка", // "cost": 1000, // "costToPay": 0, // "manualCost": 0, // "costPerUnit": 0, // "discount": 0, // "firstCost": 0, // "amount": 1 // } // ], // "date": "2023-06-03T14:33:00+00:00", // "createDate": "2023-05-31T05:51:44+00:00", // "comment": "", // "attendance": 2, // "length": 4200, // "lastChangeDate": "2023-06-02T10:26:50+00:00", // "prepaid": false, // "prepaidConfirmed": false, // "deleted": true, // "apiId": "7894" // }
```

Перенос записи на новое время: `yclients.rescheduleRecord`

Переносит запись на новое время.

Сигнатура: `yclients.rescheduleRecord(salonId: int, recordId: int, datetime: string) -> bool`

Аргументы:

- `salonId` — идентификатор филиала.
- `recordId` — идентификатор записи.
- `datetime` — новая дата и время в формате `yyyy-MM-dd HH:mm`.

Возвращаемое значение: `true` в случае успеха и `false` в случае ошибки.

Пример использования

```
$success = yclients.rescheduleRecord(25344, 52157914, "2023-06-07 16:00"); // $success  
будет true, если запись была успешно перенесена
```

Подтверждение записи: `yclients.confirmRecord`

Подтверждает запись.

Сигнатура: `yclients.confirmRecord(salonId: int, recordId: int) -> bool`

Аргументы:

- `salonId` — идентификатор филиала.
- `recordId` — идентификатор записи.

Возвращаемое значение: `true` в случае успеха и `false` в случае ошибки.

Пример использования

```
$success = yclients.confirmRecord(25344, 52157914); // $success будет true, если  
запись была успешно подтверждена
```

6.34.21. Объекты Request и Response

Методы объекта `Request`

Установка тайм-аута: `timeout`

Задаёт максимальное время выполнения запроса.

Сигнатура: `timeout(timeout: int) -> Request`

Аргументы: `timeout` — допустимое время запроса в секундах.

Пример использования:

```
$response = http.request("https://some.url", "GET").timeout(300).send();
```

Указание URL: `url`

Определяет URL для HTTP-запроса.

Сигнатура: `url(url: string) -> Request`

Аргументы: `url` — строка, представляющая адрес запроса.

Пример использования:

```
$response = http.request().url("http://some.url?p1=v1&p2=v2").method("GET").send();
```

Указание HTTP-метода: `method`

Определяет HTTP-метод (например, `GET`, `POST`, `PUT`).

Сигнатура: `method(method: string) -> Request`

Аргументы: `method` — строка с названием HTTP-метода.

Пример использования:

```
$response = http.request().url("http://some.url").method("POST").send();
```

Задание тела запроса: `body`

Определяет содержимое тела HTTP-запроса.

Сигнатура: `body(body: any) -> Request`

Аргументы: `body` — содержимое запроса.

Пример использования:

```
$response = http.request() .url("http://some.url") .method("PUT")  
.body("example body content") .send();
```

Добавление заголовка: `header`

Добавляет HTTP-заголовок к запросу.

Сигнатура: `header(header: string, value: string) -> Request`

Аргументы:

- `header` — название заголовка.
- `value` — значение заголовка.

Пример использования:

```
$response = http.request() .url("http://some.url") .method("POST") .header("Content-Type",  
"application/json") .send();
```

Указание множества заголовков: `headers`

Устанавливает набор HTTP-заголовков.

Сигнатура: `headers(headers: Map) -> Request`

Аргументы: `headers` — объект, где ключи — имена заголовков, а значения — их значения.

Пример использования:

```
$response = http.request() .url("http://some.url") .method("PUT") .headers({"Content-Type":  
"application/json", "Authorization": "Bearer token"}) .send();
```

Добавление файла: `file`

Добавляет файл для отправки в запросе.

Сигнатура:

`file(fileId: string, name: string = "") -> Request`

Аргументы:

- `fileId` — идентификатор загруженного файла.
- (Опционально) `name` — название параметра для отправки файла.

Пример использования:

```
$response = http.request() .url("http://some.url") .method("POST") .file($fileId, "uploaded_file")  
.send();
```

Отправка запроса: `send`

Отправляет запрос и возвращает объект ответа.

Сигнатура: `send() -> Response`

Пример использования:

```
$response = http.request().url("http://some.url").method("GET").send();
```

Свойства и методы объекта `Response`

Код статуса: `statusCode`

Возвращает код состояния HTTP-ответа.

Пример использования:

```
$code = $response.statusCode;
```

Тело ответа: `body`

Возвращает содержимое ответа.

Пример использования:

```
$content = $response.body;
```

Заголовки ответа: `headers`

Возвращает все заголовки ответа в виде объекта.

Пример использования:

```
$headers = $response.headers;
```

Проверка ошибки: `isError`

Определяет наличие ошибки в ответе. Возвращает `true`, если код состояния \geq 400 или есть поле `error`.

Пример использования:

```
$hasError = $response.isError();
```

Проверка успешности: `isSuccessful`

Возвращает `true`, если код состояния $<$ 400 и поле `error` отсутствует.

Пример использования:

```
$isSuccessful = $response.isSuccessful();
```

Проверка наличия заголовка: `hasHeader`

Возвращает `true`, если указанный заголовок присутствует в ответе.

Сигнатура: `hasHeader(header: string) -> bool`

Пример использования:

```
$hasContentType = $response.hasHeader("Content-Type");
```

Получение заголовка: `header`

Возвращает значение заголовка с указанным именем или пустую строку, если такого заголовка нет.

Сигнатура: `header(header: string) -> string`

Пример использования:

```
$contentType = $response.header("Content-Type");
```

Сохранение ответа в файл: `toFile`

Сохраняет тело ответа в файл и возвращает его идентификатор загруженного файла.

Пример использования:

```
$fileId = $response.toFile();
```

6.34.22. Объект `FactQuery`

Объект `FactQuery` предназначен для выполнения запросов к базе фактов. База фактов представляет собой коллекцию записей, где каждая запись (факт) имеет следующие поля:

- **Контекст** (`context`) — строка длиной до 255 символов, обозначающая предметную область. Участвует в поиске.
- **Имя факта** (`name`) — строка длиной до 255 символов, идентифицирующая факт в рамках контекста. Участвует в поиске.
- **Значение факта** (`value`) — произвольное значение, представляющее содержание факта. Поиск по этому полю не поддерживается.
- **Идентификатор бота** (`botId`) — строка, которая может быть указана для привязки факта к конкретному боту. Участвует в поиске.
- **Идентификатор клиента** (`clientId`) — строка, которая может быть указана для привязки факта к клиенту. Участвует в поиске.

Выбор полей: `select(fields)`

Задаёт список полей из базы фактов, значения которых следует вернуть в результате запроса. По умолчанию возвращаются поля `context`, `name` и `value`.

Аргументы: `fields` — строка с перечислением полей через запятую или коллекция полей, которые необходимо вернуть.

Пример использования:

```
$facts = fact.query().select("botId").rows(); // Возвращает только поле botId для всех найденных фактов.
```

Задание условия поиска: `where(field, operator, value)`

Добавляет условие поиска для фильтрации фактов. Несколько вызовов объединяются по логическому «И». Эквивалентен `andWhere`.

Аргументы:

- `field` — название поля, к которому применяется условие.
- `operator` — оператор сравнения (например, `"="`, `"~"`, `"in"`, см. доступные операторы).
- `value` — значение, с которым сравнивается поле.

Пример использования:

```
$facts = fact.query().where("context", "~", "^.example.*$").rows(); // Ищет факты, где поле "context" содержит подстроку "example".
```

Задание логического «ИЛИ»: `orWhere(field, operator, value)`

Добавляет условие поиска, объединенное с предыдущими по логическому «ИЛИ». Должно выполняться хотя бы одно из условий.

Аргументы:

- `field` — название поля, к которому применяется условие.
- `operator` — оператор сравнения (например, "=", "~", "in", см. доступные операторы).
- `value` — значение, с которым сравнивается поле.

Пример использования:

```
$facts = fact.query().where("context", "=", "test").orWhere("name", "=", "example").rows(); // Ищет факты, у которых "context" равно "test" или "name" равно "example".
```

6.34.23. Объект FactQueryCondition

Предоставляет методы для построения сложных вложенных условий при работе с базой фактов.

Задание простого условия: `where(field, operator, value)`

Добавляет простое условие к запросу. Аналогично методу `where` объекта `FactQuery`.

Сигнатура: `where(field string, operator string, value mixed)`
`FactQueryCondition`

Аргументы:

- `field` — название поля, к которому применяется условие.
- `operator` — оператор сравнения (например, "=", "~", "in", см. доступные операторы в `FactQuery`).
- `value` — значение для сравнения с полем.

Пример использования:

```
$condition = fact.cond().where("context", "~", "^.example.*$"); // Условие: поле "context" содержит подстроку "example".
```

Логическое «И» для простого условия: `andWhere(field, operator, value)`

Добавляет простое условие, объединенное с предыдущими условиями по логическому «И». Аналогично методу `andWhere` объекта `FactQuery`.

Сигнатура: `andWhere(field string, operator string, value mixed)`
`FactQueryCondition`

Аргументы:

- `field` — название поля, к которому применяется условие.
- `operator` — оператор сравнения.

- `value` — значение для сравнения с полем.

Пример использования:

```
$condition = fact.cond().where("context", "~", "^.*example.*$").andWhere("name", "=", "test"); // Условия: "context" содержит "example" И "name" равно "test".
```

Логическое «ИЛИ» для простого условия: `orWhere(field, operator, value)`

Добавляет простое условие, объединенное с предыдущими условиями по логическому «ИЛИ». Аналогично методу `orWhere` объекта `FactQuery`.

Сигнатура: `orWhere(field string, operator string, value mixed) FactQueryCondition`

Аргументы:

- `field` — название поля, к которому применяется условие.
- `operator` — оператор сравнения.
- `value` — значение для сравнения с полем.

Пример использования:

```
$condition = fact.cond().where("context", "=", "example").orWhere("name", "=", "test"); // Условия: "context" равно "example" ИЛИ "name" равно "test".
```

Вложенные условия: `where(cond)`

Добавляет вложенное условие. Аналогично методу `where(cond)` объекта `FactQuery`.

Сигнатура: `where(cond FactQueryCondition) FactQueryCondition`

Аргументы: `cond` — объект `FactQueryCondition`, представляющий вложенное условие.

Пример использования:

```
$nestedCondition = fact.cond().where("name", "=", "example").orWhere("name", "^@", "test"); $condition = fact.cond().where("context", "~", "^.*example.*$").andWhere($nestedCondition); // Условие: "context" содержит "example" И ("name" равно "example" ИЛИ начинается с "test").
```

Логическое «И» для вложенного условия: `and(cond)`

Добавляет вложенное условие, объединенное с предыдущими по логическому «И». Аналогично методу `andWhere(cond)` объекта `FactQuery`.

Сигнатура: `and(cond FactQueryCondition) FactQueryCondition`

Аргументы: `cond` — объект `FactQueryCondition`, представляющий вложенное условие.

Пример использования:

```
$nestedCondition = fact.cond().where("name", "=", "example"); $condition = fact.cond().where("context", "~", "^.*example.*$").and($nestedCondition); // Условие: "context" содержит "example" И ("name" равно "example").
```

Логическое «ИЛИ» для вложенного условия: `or(cond)`

Добавляет вложенное условие, объединенное с предыдущими по логическому «ИЛИ». Аналогично методу `orWhere(cond)` объекта `FactQuery`.

Сигнатура: `or(cond FactQueryCondition) FactQueryCondition`

Аргументы: `cond` — объект `FactQueryCondition`, представляющий вложенное условие.

Пример использования:

```
$nestedCondition = fact.cond().where("name", "=", "example"); $condition = fact.cond().where("context", "~", "^.*example.*$").or($nestedCondition); // Условие: "context" содержит "example" ИЛИ ("name" равно "example").
```

6.34.24. Объект `UserMessage`

Объект `UserMessage` используется для работы с сообщениями, отправленными пользователем, включая текст и вложения.

Текст сообщения: `message`

Возвращает оригинальный текст сообщения.

Сигнатура: `message string`

Аргументы: отсутствуют.

Пример использования:

```
$msg = queue.last().message; // $msg содержит текст последнего сообщения пользователя.
```

Вложения сообщения: `attachments`

Возвращает список идентификаторов файлов, приложенных к сообщению.

Сигнатура: `attachments List<string>`

Аргументы: отсутствуют.

Пример использования:

```
$attachments = queue.first().attachments; // $attachments содержит список вложений первого сообщения пользователя.
```

Проверка пустого сообщения: `isEmpty()`

Определяет, пустое ли сообщение.

Сигнатура: `isEmpty() bool`

Аргументы: отсутствуют.

Возвращаемое значение:

- `true` — если сообщение пустое.
- `false` — если сообщение содержит текст или вложения.

Пример использования:

```
$isEmpty = queue.last().isEmpty(); // $isEmpty содержит true, если последнее сообщение пользователя пустое, и false в противном случае.
```

Проверка наличия вложений: `hasAttachments()`

Определяет, есть ли вложения в сообщении.

Сигнатура: `hasAttachments() bool`

Аргументы: отсутствуют.

Возвращаемое значение:

- `true` — если вложения присутствуют.
- `false` — если вложений нет.

Пример использования:

```
$hasAttachments = queue.first().hasAttachments(); // $hasAttachments содержит true, если первое сообщение пользователя имеет вложения, и false в противном случае.
```

6.34.25. Объект Sentence

Объект `Sentence` представляет данные, полученные после обработки текста Natural Language Understanding (NLU).

Распознанное намерение: `intent`

Возвращает название распознанного намерения.

Сигнатура: `intent string`

Пример использования:

```
$sentence = nlu.parse("Привет Вася", "d926726a-5acb-4233-8c1e-ce4300921de0");  
$intent = $sentence.intent; // $intent содержит "greeting".
```

Достоверность намерения: `intentConfidence`

Определяет степень достоверности распознанного намерения (1 – максимальная достоверность, 0 – минимальная).

Сигнатура: `intentConfidence number`

Пример использования:

```
$sentence = nlu.parse("Привет Вася", "d926726a-5acb-4233-8c1e-ce4300921de0");  
$confidence = $sentence.intentConfidence; // $confidence содержит 0.98.
```

Список намерений: `intents`

Возвращает список всех распознанных намерений. Каждый элемент — кортеж из двух элементов:

- Название намерения (string).
- Достоверность намерения (float).

Сигнатура: `intents List<Tuple>`

Пример использования:

```
$sentence = nlu.parse("Доброе утро Вася, пора вставать", "d926726a-5acb-4233-8c1e-ce4300921de0"); $intents = $sentence.intents; // $intents содержит [("greetings", 0.97), ("wakeup", 0.88)]. $first = $intents.get(0); // $first содержит ("greetings", 0.97). $intent = $first.get(0); // $intent содержит "greetings".
```

Список сущностей: `entities`

Возвращает список распознанных сущностей. Каждый элемент — кортеж из трех элементов:

- Тип сущности (string).
- Значение сущности (string).
- Достоверность распознавания сущности (float).

Сигнатура: `entities List<Tuple>`

Пример использования:

```
$sentence = nlu.parse("Доброе утро Вася", "d926726a-5acb-4233-8c1e-ce4300921de0"); $entities = $sentence.entities; // $entities содержит [("human-name", "Вася", 0.96), ("time", "2023-01-09 23:30:00", 0.87)]. $first = $entities.get(0); // $first содержит ("human-name", "Вася", 0.96). $type = $first.get(0); // $type содержит "human-name".
```