ASTERCONF
- 2019

ASTERISK + TINKOFF VOICEKIT

И ПОЧЕМУ МЫ ОТКАЗАЛИСЬ ОТ MRCP

Окопник Григорий

Tinkoff.ru

Разработчик
отдела Голосовых Технологий

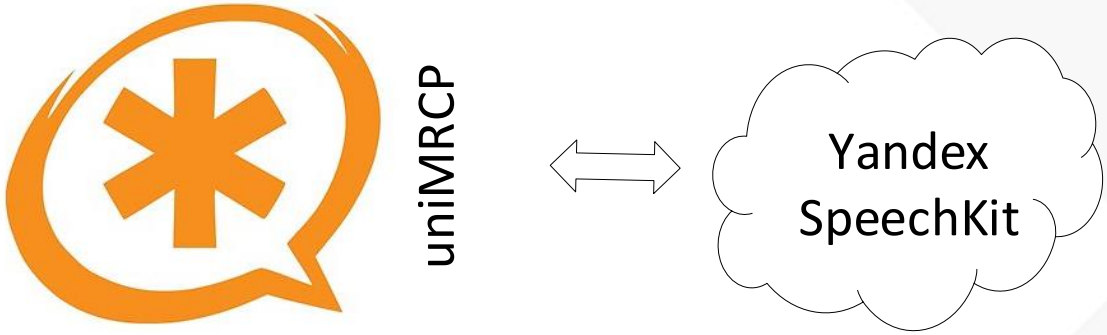
О чём доклад?

- Зачем вообще строить роботов с распознаванием и синтезом речи.
- Что не так с доступными интеграциями.
- Наше Open Source решение для использования распознавания и синтеза в Asterisk.
- Как вы сможете построить своих роботов уже сегодня.

Мотивация делать роботов

- Телефонные разговоры – один из основных каналов взаимодействия с клиентом
- Доля простых разговоров достигает **85%**
- Простые разговоры легко автоматизируются
- Конверсия в целевой результат **2%**

Первое решение (MVP)



- На тот момент не устраивало качество распознавания.
- Есть собственные сервисы распознавания и синтеза речи: Tinkoff VoiceKit.
- Ограниченная интеграция с Asterisk.
- Решили делать свою интеграцию.

Как интегрировать сервисы?

- Asterisk Speech Recognition API?
- Использовать модули uniMRCP?
- Интегрироваться через EAGI?
- Разработать свои модули для Asterisk?

Интеграция через uniMRCP

- Используется блокирующая логика (подобно Asterisk Speech Recognition API): приложение `SynthAndRecog()` диктует текст и распознаёт одну фразу.
- В архитектуре нет сочетаемости логики: `Synth()`, `Recog()` и `SynthAndRecog()` – 3 отдельных приложения
- Распознавание прерывается при получении фразы.
- Нет возможности прервать синтез.
- Нет работы с промежуточными гипотезами.
- Нет возможности удобно комбинировать синтез и проигрывание файлов.

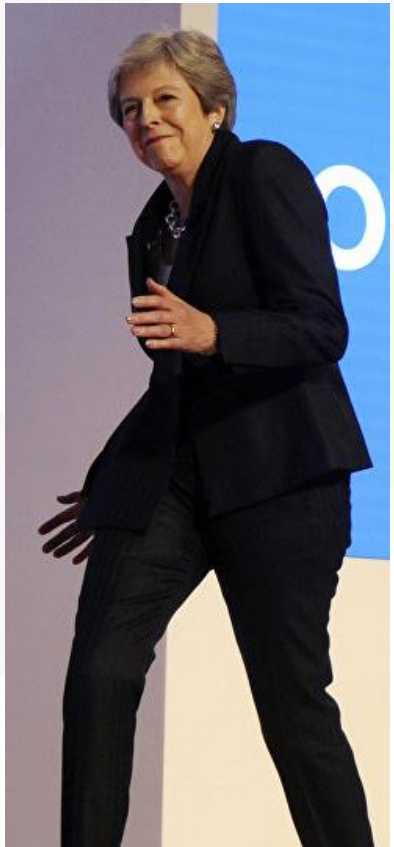
Интеграция через EAGI

- Нет ограничений по возможностям
- Не создаётся переиспользуемой компоненты для Asterisk
- Asterisk используется как proxy
- Нет привязки к языку – нужно сформировать культуру разработки
- При каждом вызове EAGI создаётся новый процесс
- Управление звонками доступно
  только при вызове обработчика EAGI (в отличие от AMI или ARI)

Проектируем свои модули

Модули для использования
распознавания и синтеза речи в Asterisk:

- Разделение функций.
- Сочетаемость.
- Неблокирующая архитектура:
  как неблокирующие сокеты в POSIX.
- Переиспользуемость.
- Какой протокол взять за основу?



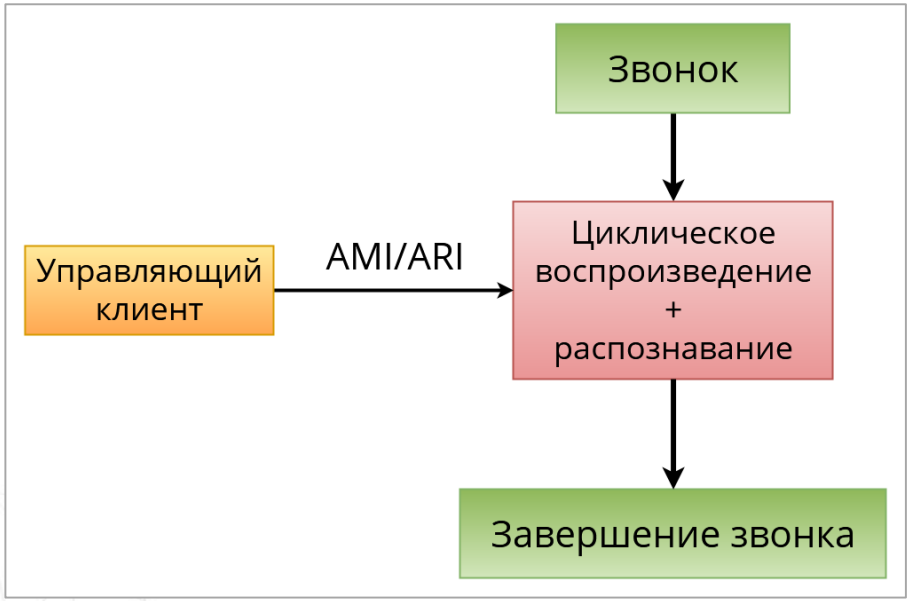
Почему мы не используем MRCP?

- В Tinkoff VoiceKit реализовано API на базе gRPC: HTTP2 + Protobuf
- Нечего переиспользовать: пришлось бы дорабатывать модули uniMRCP и добавлять поддержку MRCP в сервисы.
- Неоптимальное сетевое взаимодействие:
  - Несжатые заголовки – наследие HTTP/1.1
  - Оверхеды API в стиле REST.
    Например, промежуточные гипотезы отдаются по запросу: получаем ответы с задержкой и лишние запросы.

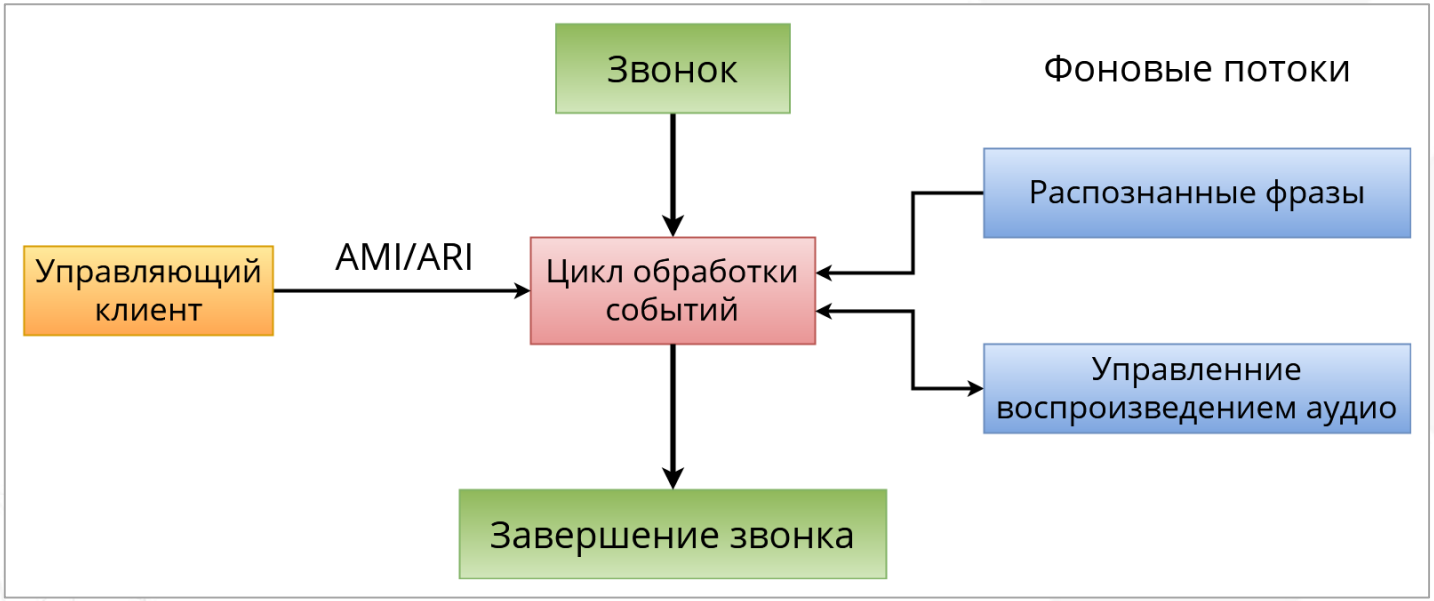
Модули Tinkoff AstVoiceKit

- app_grpcsttbackground.so
  - GRPCSTTBackground() – запуск распознавания в фоне
- app_playbackground.so
  - PlayBackground() – управление воспроизведением и синтезом
  - PlayBackgroundInitGRPCTTS() – инициализация сессии синтеза
- app_waitevent.so
  - WaitEvent() – получение события по таймауту
  - WaitEventInit() – инициализация очереди событий
- func_gettimensec.so
  - ${GET_TIME_NSEC()} – функция получения времени

Логика в dialplan: uniMRCP



Логика в dialplan: AstVoiceKit



uniMRCP в AEL

```
uniMRCP => {
    Answer();

    while (/* Максимальное время разговора не истекло */) {
        SynthAndRecog(text,grammar,options);
        /*
          Обработка распознанного текста
         */
    }
}
```

AstVoiceKit в AEL

```
AstVoiceKit => {
    Answer();
    // Инициализация распознавания и синтеза

    while (/* Максимальное время разговора не истекло */) {
        WaitEvent(/* Время до ближайшего события (по таймеру) */);
        switch (${WAITEVENTNAME}) {
            // Обработка:
            // - событий распознавания
            // - событий воспроизведения
            // - управляющих команд
        }
    }
}
```

AstVoiceKit: распознавание речи

`app_grpcsttbackground.so`

- Для включения нужно запустить приложение `GRPCSTTBackground()`.
- Модуль получает фрэймы в фоне.
- Кто-то должен вычитывать фрэймы.
- Модуль не ворует фрэймы (можно записывать входящее аудио).

AstVoiceKit: воспроизведение аудио

`app_playbackground.so`

Можно было бы использовать встроенные приложения Playback() и Background(), но они блокируют управление

- Очередь воспроизведения обрабатывается в фоне.
- Приложение `PlayBackground()` отправляет в очередь: файлы для проигрывания или текст для синтеза.
- Для использования синтеза нужно предварительно инициализировать соединение приложением `PlayBackgroundInitGRPCTTS()`.
- Есть возможность накладывать аудио в 4 слоя.
  Например, для добавления фоновой музыки.

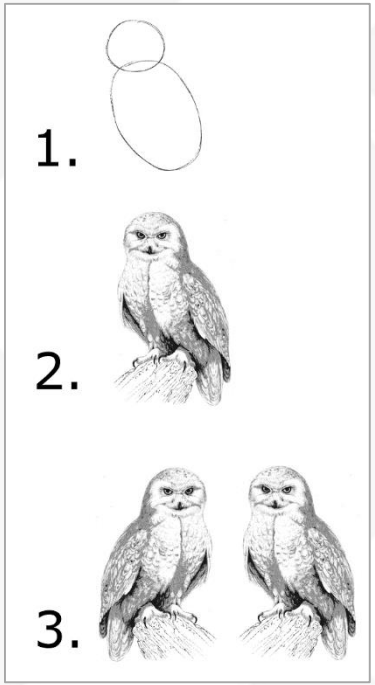
AstVoiceKit: управление событиями

`app_waitevent.so`

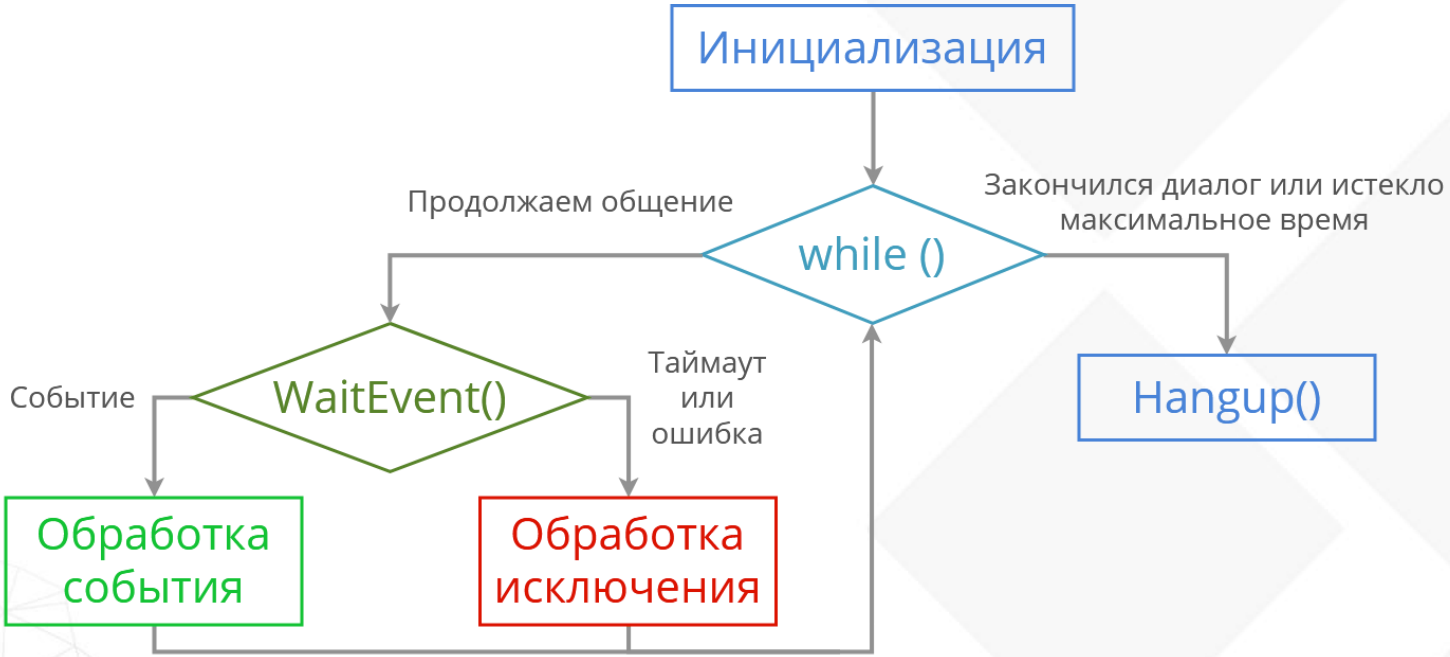
- Для сбора событий нужно инициализировать очередь приложением `WaitEventInit()`.
- Для обработки событий нужно вычитывать события в цикле приложением `WaitEvent()`.
- `WaitEvent()` возвращает одно событие за раз:
  - текст распознанной речи
  - статус проигрывания аудио или синтеза текста
  - управляющую команду, отправленную через AMI или ARI клиента (проиграть файл, синтезировать текст, переключить на оператора и т. п.)
- Для генерации событий (как правило для AMI или ARI клиента) используется встроенное приложение `UserEvent()`.

Роботостроение в примерах

- **Шаг 1: Простой опросник**
  - Задаём один вопрос
  - Получаем ответ, пишем ответ в лог
- **Шаг 2: Сложный опросник**
  - Ведём сложную диалоговую логику
  - Обрабатываем события в AMI клиенте
  - Управление через AMI
- **Шаг 3: Добавляем синтез**
  - Расширяем управление через AMI



Общая логика в dialplan



Шаг 1: Простой опросник

```
// Инициализируем распознавание и обработку событий
Answer();
WaitEventInit();
GRPCSTTBackground();

// Приветствие
Playback(greeting_and_question);
…
// [Получаем текст распознанной речи (ждём 5 секунд)]
…

// Прощание
Playback(bye);
Hangup();
```

Шаг 1: Простой опросник

```
// Получаем распознанный текст (ждём 5 секунд):
Set(CALL_END_TIME=$[${GET_TIME_NSEC(MONOTONIC)} + 5]);
Set(SLEEP_TIME=$[${CALL_END_TIME} - ${GET_TIME_NSEC(MONOTONIC)}]);

while (${SLEEP_TIME} > 0) {
    WaitEvent(${SLEEP_TIME});
    if (${WAITEVENTSTATUS} == SUCCESS) {
        switch (${WAITEVENTNAME}) {
        case GRPCSTT_UTF8:
            Log(NOTICE,Client response:${WAITEVENTBODY});
            Set(SLEEP_TIME=0);
            break;
        }
    }
    Set(SLEEP_TIME=$[${CALL_END_TIME} - ${GET_TIME_NSEC(MONOTONIC)}]);
}
```

Шаг 1: Простой опросник

Шаг 2: Сложный опросник

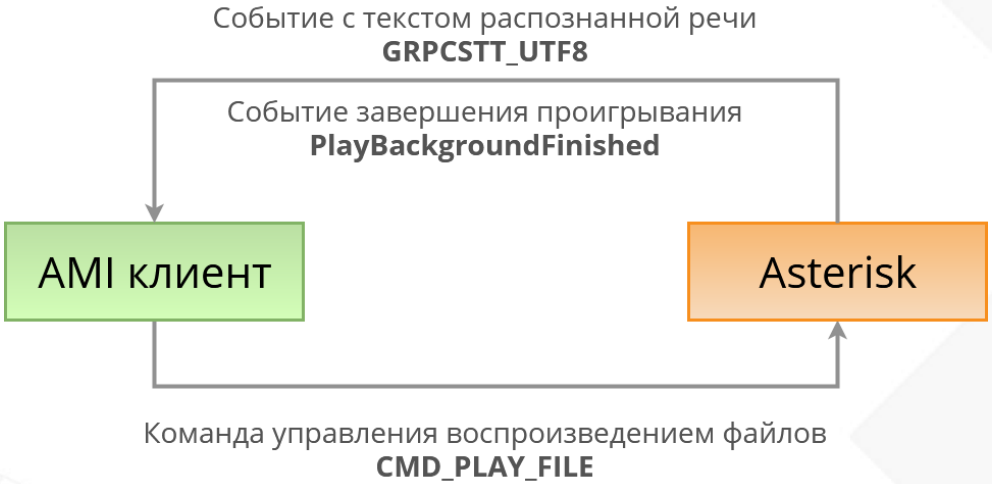
```
  Answer();
  WaitEventInit();
  GRPCSTTBackground();

- Playback(greeting);

  ...
  // получаем текст
  ...

- Playback(bye);
  Hangup();
```

Шаг 2: Сложный опросник

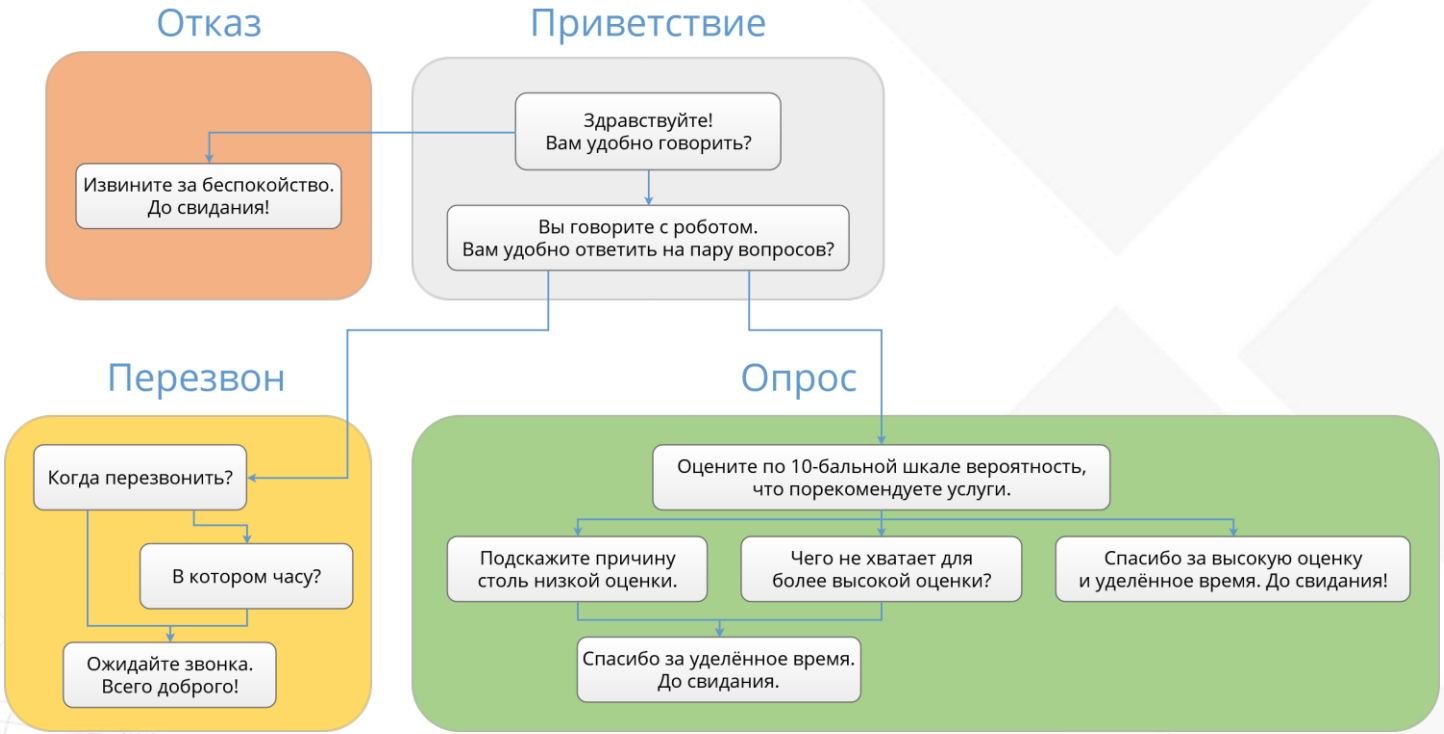
```
  …
  while (${SLEEP_TIME} > 0) {
      WaitEvent(${SLEEP_TIME});

      if (${WAITEVENTSTATUS} == SUCCESS) {
          switch (${WAITEVENTNAME}) {
-         case GRPCSTT_UTF8:
-             Log(NOTICE,Client response:${WAITEVENTBODY});
-             Set(SLEEP_TIME=0);
-             break;
+         case CMD_PLAY_FILE:
+             PlayBackground(play,,${WAITEVENTBODY});
+             break;
          }
      }
      Set(SLEEP_TIME=$[${CALL_END_TIME} - ${GET_TIME_NSEC(MONOTONIC)}]);
  }
```

Шаг 2: Сложный опросник

Добавляем AMI клиент



Шаг 2: Сложный опросник



Шаг 3: Добавляем синтез

```
  Answer();
  WaitEventInit();
  GRPCSTTBackground();
+ PlayBackgroundInitGRPCTTS();

  …
  // получаем текст
  …

  Hangup();
```

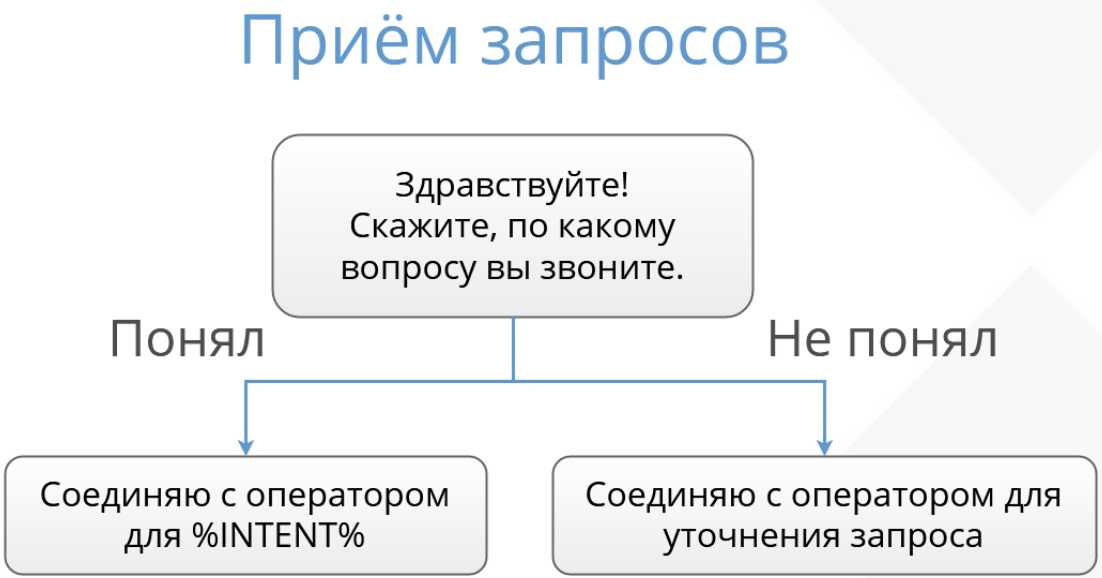
Шаг 3: Добавляем синтез

```
  while (${SLEEP_TIME} > 0) {
      WaitEvent(${SLEEP_TIME});

      if (${WAITEVENTSTATUS} == SUCCESS) {
          switch (${WAITEVENTNAME}) {
-         case CMD_PLAY_FILE:
-             PlayBackground(play,,${WAITEVENTBODY});
-             break;
+         case CMD_PLAY:
+             PlayBackground(${WAITEVENTBODY});
+             break;
          }
      }
      Set(SLEEP_TIME=$[${CALL_END_TIME} - ${GET_TIME_NSEC(MONOTONIC)}]);
  }
```

Шаг 3: Добавляем синтез

Расширяем AMI клиент

Шаг 3: Добавляем синтез



Смотрим в сегодняшний день

- Несколько типов роботов на базе интеграции Asterisk + VoiceKit в работе.
- Модули Tinkoff AstVoiceKit в Open Source:
  https://github.com/TinkoffCreditSystems/asterisk-voicekit-modules
- Список поддерживаемых сервисов расширяется:
  в планах интеграция с Yandex SpeechKit и Google Cloud Speech.
- Сейчас можно собрать свою интеграцию через сервисы Tinkoff VoiceKit.
- Раздаём тестовые ключи для использования VoiceKit для всех желающих.
  https://voicekit.tinkoff.ru/ – форма заявки на получение ключа.

СТРОЙТЕ РОБОТОВ ВМЕСТЕ С НАМИ!

Григорий Окопник

g.e.okopnik@tinkoff.ru

ASTERCONF - 2019