Настройка интеграции iAStream с 1С

Введение

Платформа 1С в России является одной из популярных систем автоматизации управления предприятием. В связи с этим существует необходимость интеграции специализированных систем с данной платформой, в том числе и систему распознавания автомобильных номеров iAStream.

В данной работе рассматривается вариант взаимодействия этих подсистем через http-протокол.

Интеграция заключается передачи сигналов о распознанных событиях из системы iAstream в 1С на основе механизма http-сервисов. Моделируется ситуация автоматического формирования нарядов на мойке в результате распознавания номеров авто при въезде и выезде.

Обобщенный алгоритм интеграции

Алгоритм обработки от iAStream включает следующие основные шаги:

- сигнал в виде http-запроса от iAControl поступает в Apache-сервер, который вызывает модуль 1С;
- модуль 1С ищет заданную функцию обработки запроса и исполняет его.

Алгоритм интеграции включает следующие шаги:

- установку необходимого программного обеспечения;
- конфигурирование информационной базы 1С и реализацию функции обработки запроса;
- настройка привязки 1С и Apache механизмом «Публикация сервиса на веб-сервере»;
- тестирование системы имитационными сигналами от браузера;
- настройка iAStream для передачи сигналов в 1С.

Список используемых программных средств

Для демонстрации возможности интеграции в 1С использовалось следующее программное обеспечение:

- платформа 1С 8.3 (в данном примере использовалась 32- разрядная учебная версия);
- веб-сервер — Apache (тоже 32 разрядная, 64-разрядная не работала с 32-разрядной 1С);
- программа для автоматического распознавания автомобильных номеров и отправки сигналов iAStream.

Установка и настройка Apache

Для подготовки веб-сервера для процесса интеграции необходимо выполнить следующие шаги:

1. Готовую сборку под Windows можно скачать со следующего сайта: https://www.apachelounge.com/.

2. Распаковываем архив Apache. Например, по следующем пути: C:\Apache24.

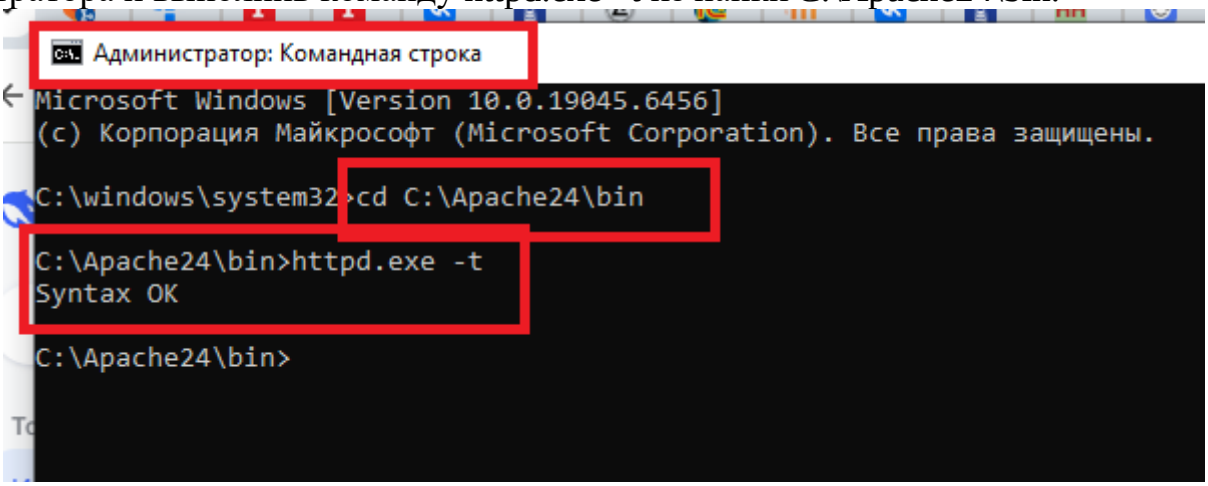
3. Настраиваем конфигурационный файл Apache httpd.conf (в папке C:\Apache24\conf):

   a. Настройка параметра **ServerRoot**: директива ServerRoot должна указывать на папку установки Apache, например, C:/Apache24.

   b. Настройка порта определяется директивой **Listen** 80. Если порт занят, то изменяем его, например, на **Listen** 5000. В данном примере используется порт 5000.

   c. Настройка параметра **ServerName**: localhost:5000.

4. Проверяем конфигурацию, запустив командную строку с правами администратора и выполнив команду httpd.exe –t из папки C:\Apache24\bin.



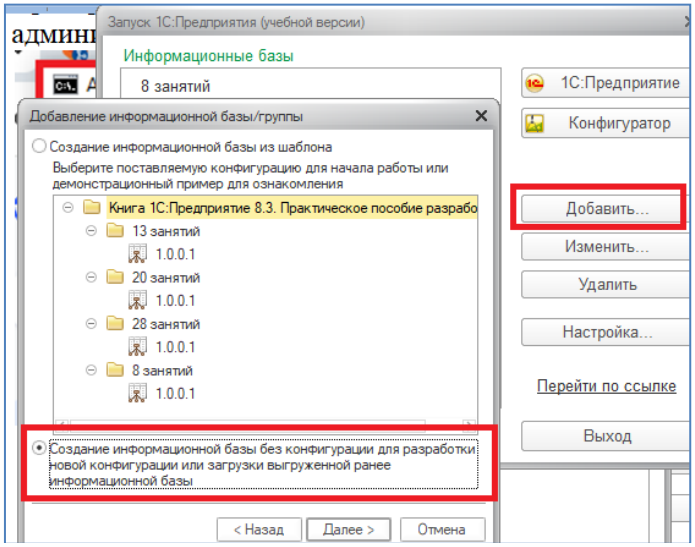
5. Устанавливаем Apache как служба командой httpd.exe -k install. Если при этом появится запрос от брандмауэра Windows, разрешаем доступ для частных и публичных сетей.

6. Запуск службы можно осуществить командой httpd.exe -k start.

7. Для проверки работы сервера достаточно набрать в браузере адрес http://localhost:5000. При отсутствии ошибок считаем, что Apache установлен успешно.

Настройка конфигурации 1С

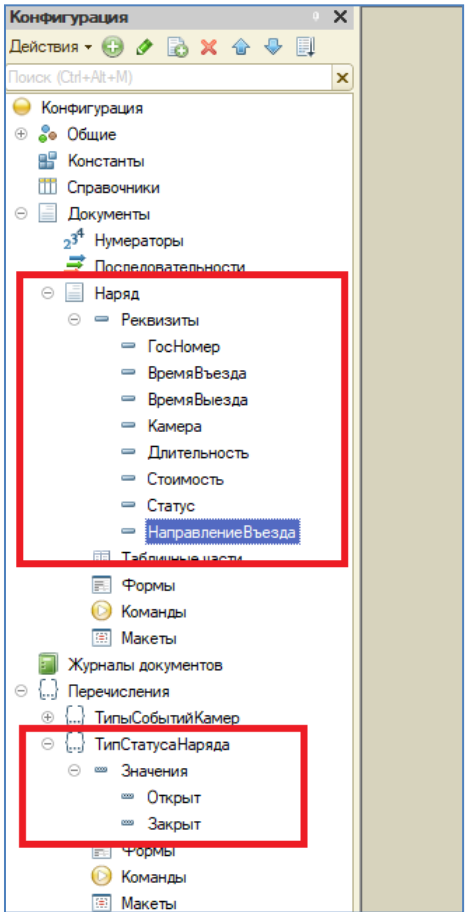
1. Для демонстрации создадим новую информационную базу. При создании информационной базы выбираем имя, например, «Камера». Остальные настройки могут остаться по умолчанию.



.

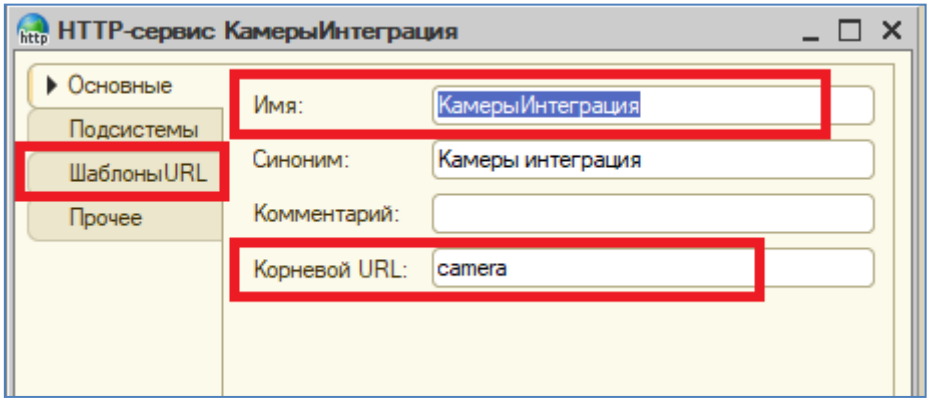
2. Заходим в Конфигуратор и определяем структуру данных: добавляем один документ «Наряд» со следующими реквизитами: ГосНомер(строка), ВремяВъезда(дата и время), ВремяВыезда(дата и время), Камера(число), Длительность(число), Статус (Перечисление.ТипСтатусаНаряда), НаправлениеВъезда (число).

Перечисление.ТипСтатусаНаряда может принимать значения «Открыт» и «Закрыт».

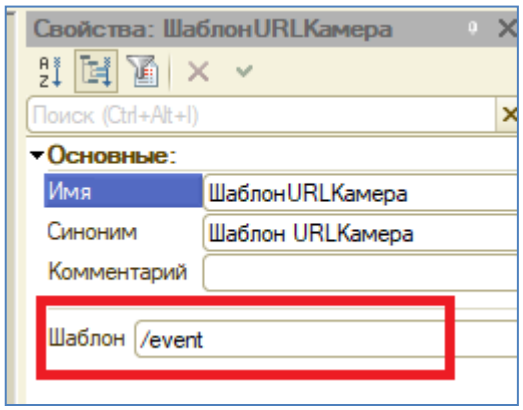


3. Создаем HTTP-сервис, который будет вызываться на запрос от iAStream. Для этого выполняем следующие действия:

- в дереве метаданных выбираем пункт меню «Общие → HTTP-сервисы → Добавить новый». Имя «Камера Интеграция, значение параметра корневой URL – «camera».



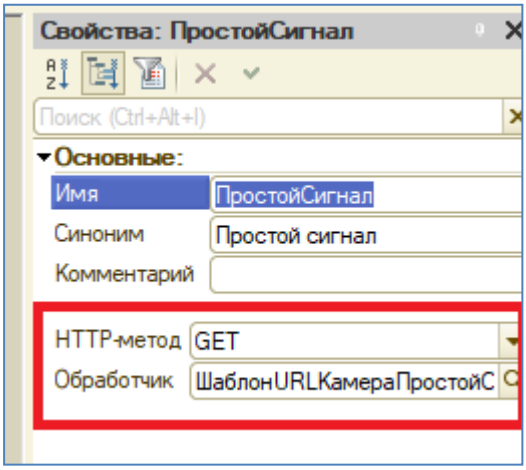
- создаём шаблон URL, выбрав в свойствах "ШаблоныURL"/ В свойствах Шаблон укажем значение: «/event».



- добавим метод для шаблона. В данной демонстрации используется метод GET (простая передача параметров в URL):

- HTTP-метод: GET;
- обработчик: нажмите на значок справа. В открывшемся редакторе наберите код, который будет обрабатывать запрос. В результате будет создана функция:

```
Функция ШаблонURLКамераПростойСигнал(Запрос)
КонецФункции
```



4. Реализация функции обработчика представлена в Приложении А. Дополнительные реализованные функции в общих модулях, такие как логирование сообщений в файл, создание документа наряда и другие представлены в приложениях Б и В.

5. Учебная версия имеет ограничения по количеству версий для запуска 1С сервера и при работе веб-сервера приходится закрывать конфигуратор и 1С Предприятие, что усложняет процесс отладки. В связи этим реализована процедура «ЗаписатьВЛог» для записи отладочных сообщений в текстовый файл в общем модуле «ЛоггированиеКамер». Реализация представлена в приложении Б.

Описание основных моментов функции HTTP-сервиса

Функция HTTP-сервиса вызывается веб-сервером и выполняет следующие основные функции:

- обработка параметры HTTP-запроса;
- выполнение действий согласно задачам бизнес-логики;
- формирование ответа в соответствии с протоколом HTTP.

Пример кода функции обработчика представлен в приложении А. Ниже представлено пояснение основных моментов:

1. Извлечение параметров из HTTP-запроса. Платформа 1С при обработке запросов создает специальный объект, в который заполняет результаты распаковки параметров HTTP-протокола, и передает в качестве параметра в функцию обработчика. Ниже приведен пример извлечения параметров запроса:

```
Функция ШаблонURLКамераПростойСигнал(Запрос)
        Параметры = Запрос.ПараметрыЗапроса;
        // Извлекаем нужные значения
        StreamIdx = Параметры.Получить("stream");
        Direction = Параметры.Получить("dr");
        DateTimeStr = Параметры.Получить("date");
        Plate = Параметры.Получить("plate");
```

2. Далее осуществляем предварительную проверку корректности параметров запроса и при возможности исправление их. Проверяется, что параметр stream является числом. При неправильности значения данного параметра при помощи функции СоздатьОтветОшибки(описание кода функции приведено в приложении А) возвращается ответ об ошибки. Также содержимое переменной Direction из текста преобразуется в числовой код:

```
Попытка
        StreamIdx =Число(StreamIdx);
Исключение
        ЛоггированиеКамер.ЗаписатьВЛог("Ошибка: " + "Номер камеры должен быть числом" , "INFO");
        Возврат СоздатьОтветОшибки(400, "Номер камеры должен быть числом");
КонецПопытки;
ЛоггированиеКамер.ЗаписатьВЛог("Направление =" + Direction , "INFO");
Если  Direction="OUT" Тогда
         Direction = 1;
 Иначе
         Direction = 0;
КонецЕсли;
```

3. Отсутствие обязательных параметров StreamIdx и plate аналогично возвращает ошибку:

```
// Проверка обязательных параметров
```

```
    Если StreamIdx = Неопределено ИЛИ Plate = Неопределено Тогда
      Возврат СоздатьОтветОшибки(400,"Ошибка: не указаны stream или plate");
КонецЕсли;
```

4. Регистрация времени сигнала осуществляется на основе параметра date HTTP-запроса. При невозможности извлечь корректные данные, берется текущее время и осуществляется запись в логе с предупреждением о данной ситуации. Функция ПреобразоватьНестандартнуюДату реализована в общем модуле КамерыСервер и представлена в приложении В:

```
// Преобразуем дату из строки
ВремяСобытия = ТекущаяДата(); // по умолчанию
Если DateTimeStr <> Неопределено Тогда
  Попытка
    ВремяСобытия = КамерыСервер.ПреобразоватьНестандартнуюДату(DateTimeStr);
  Исключение
    // Если не удалось преобразовать, оставляем текущую
                ЛоггированиеКамер.ЗаписатьВЛог("Ошибка" + ОписаниеОшибки() , "INFO");
  КонецПопытки;
КонецЕсли;
```

5. Осуществляем попытку найти наряд со статусом открыт для данного номера и идентификатора камеры. Функция НайтиОткрытыйНаряд реализована в общем модуле КамерыСервер:

```
    //ищем открытый наряд
    СуществующийНаряд = КамерыСервер.НайтиОткрытыйНаряд(Plate, StreamIdx);
        //если это новый наряд
```

6. Если наряд не существует, то создаем новый с помощью функции СоздатьНарядНаВъезд. В случае обнаружения открытого наряда сперва проверяем параметр НаправлениеВъезда документа. Если направление въезда текущего сигнала и найденного открытого наряда совпадает, то считаем, что это повторный запрос. В обратном случае - осуществляется закрытие наряда с предварительным длительности нахождения транспортного средства в мойке и расчетом стоимостиуслуги:

```
    Если СуществующийНаряд = Неопределено Тогда
        //создаем новый наряд
        НовыйНаряд = КамерыСервер.СоздатьНарядНаВъезд(Plate, StreamIdx, ВремяСобытия,
Direction);
        Результат.Вставить("Наряд", Строка(НовыйНаряд));
        Результат.Вставить("Статус", "Новый наряд");
        Результат.Вставить("Код", 0);
        ЛоггированиеКамер.ЗаписатьВЛог("Создан наряд " +  строка(НовыйНаряд), "INFO");
    Иначе
        //повторный сигнал = игнорируем
        Если СуществующийНаряд.НаправлениеВъезда = Direction Тогда
            Результат.Вставить("Наряд", Строка(СуществующийНаряд));
            Результат.Вставить("Статус", "Существует наряд");
            Результат.Вставить("Код", 1);
            ЛоггированиеКамер.ЗаписатьВЛог("Наряд существует " +  СуществующийНаряд,
"INFO");
        Иначе
            //р
            НарядОбъект = СуществующийНаряд.ПолучитьОбъект();
            РезультатРасчета =
РасчетСтоимости.РассчитатьСтоимость(НарядОбъект.ВремяВъезда,ВремяСобытия);
            НарядОбъект.ВремяВыезда = ВремяСобытия;
        НарядОбъект.Длительность = РезультатРасчета.ДлительностьМинут;
            НарядОбъект.Стоимость = РезультатРасчета.Стоимость;
```

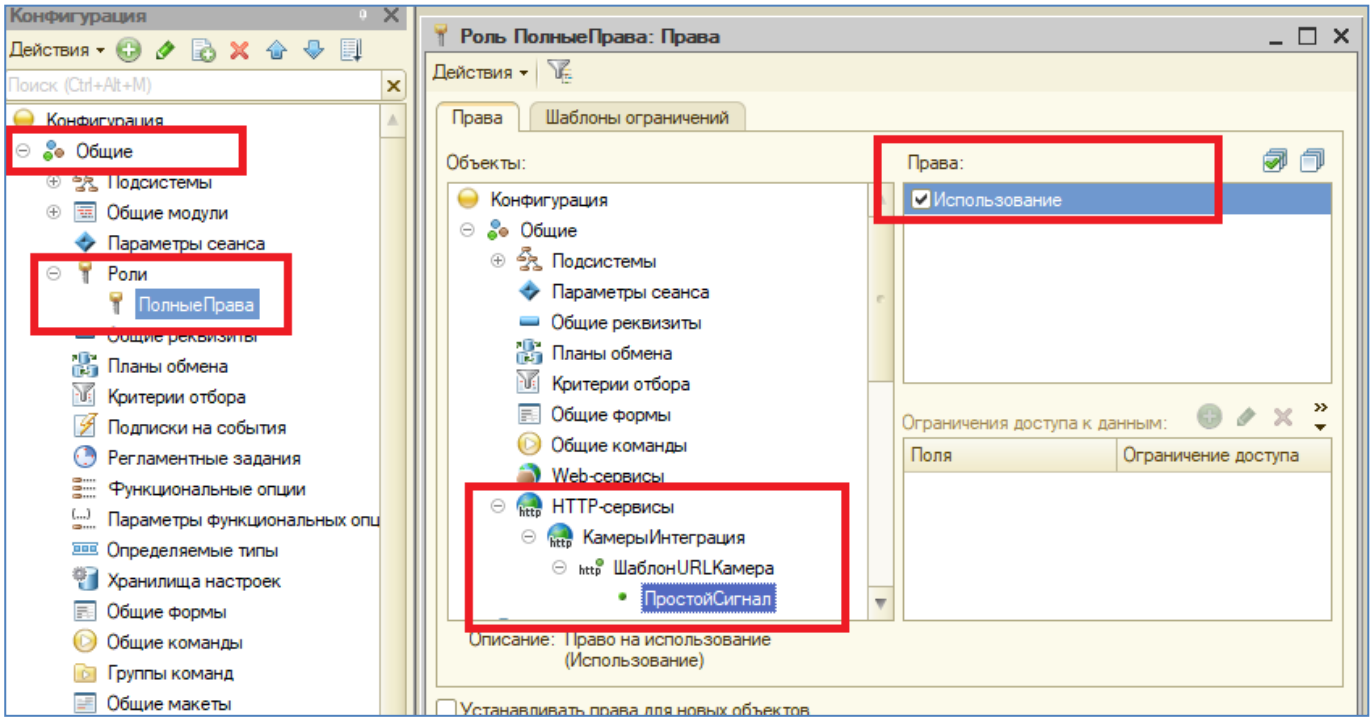
```
                НарядОбъект.Статус = Перечисления.ТипСтатусаНаряда.Закрыт;
                НарядОбъект.Записать();
                        Результат.Вставить("Наряд", Строка(СуществующийНаряд));
                        Результат.Вставить("Статус", "Закрытие наряда");
                        Результат.Вставить("Код", 2);
                        ЛоггированиеКамер.ЗаписатьВЛог("Закрытие наряда " +
Строка(СуществующийНаряд), "INFO");
                КонецЕсли

        КонецЕсли;
        Возврат СоздатьОтветУспеха(Результат);
КонецФункции
```

7. В конце возвращается ответ на запрос из функции СоздатьОтветУспеха.

Создание пользователя в 1С

Для обеспечения аудита, контроля и разграничения прав доступа создадим отдельного пользователя, от имени которого будут обрабатываться сигналы от iAStream. Для этого выполним следующие шаги:

1. Заходим в базу в режиме конфигуратора.
2. Переходим в пункт Администрирование| Пользователи. Добавляем нового пользователя. Выставляем галочку «Аутентификация 1С: Предприятия». Ограничения учебной версии не позволяют выставить пароль.
3. На вкладке «Прочие» добавляем доступные роли «Полные права». И сохраняем параметры пользователя.
4. Переходим в раздел конфигурации «Общие | Роли | Полные права». Проверяем, что роль «Полные права» имеет право использование http-сервиса (конкретно Get- сигнала)



5. Также проверяем права доступа к документу «Наряд».

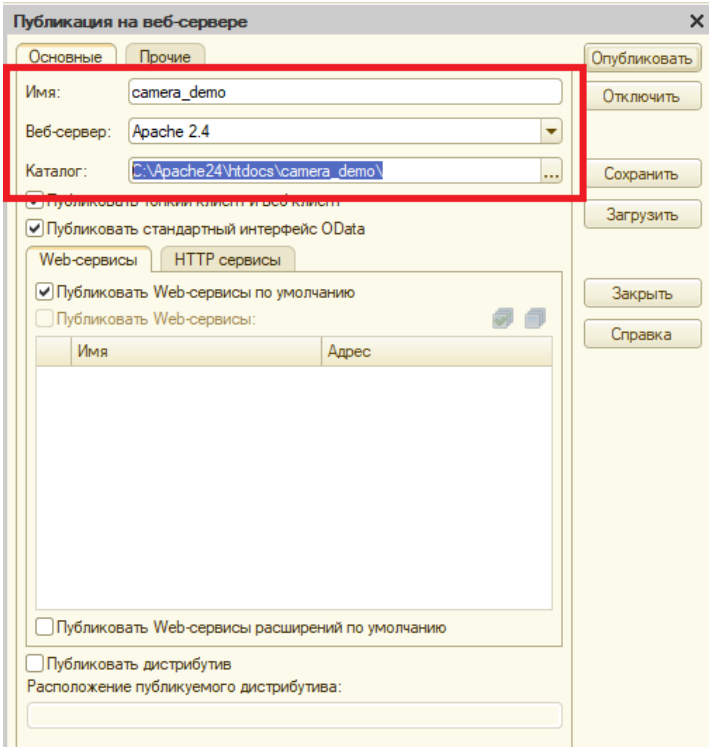
Данных действий достаточно для регистрации пользователя для обработки сигналов от iAStream.

Публикация HTTP-сервиса

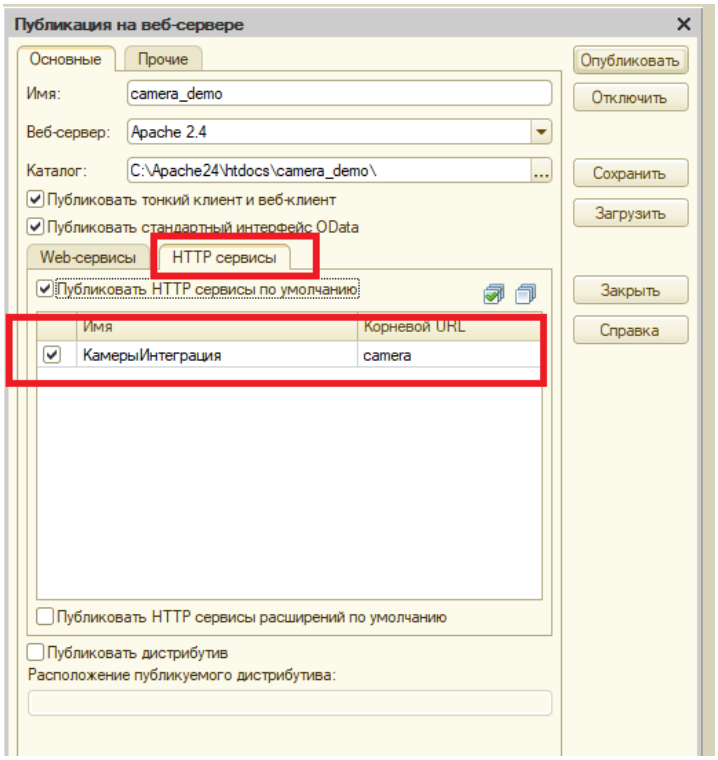
Для возможности вызова функции 1С веб-сервером требуется выполнить процедуру публикации HTTP-сервиса. Для этого необходимо:

1. В конфигураторе выбрать пункт меню «Администрирование| Публикация на веб-сервере».
2. В появившемся окне введите:

- «Имя» - camera_demo;
- «Веб-сервер» -Apache2.4;
- «Каталог» - C:\Apache24\htdocs\camera_demo\. C:\Apache24 – это путь к установленному серверу. Папка htdocs существует по умолчанию. Каталог camera_demo – нужно создавать.



3. На закладке HTTP-сервисы ставим галочку напротив нашего сервиса.



4. Нажимаем кнопку «Опубликовать».

Тестирование HTTP-сервиса.

Тестирование через браузер

Для тестирования на первоначальном этапе можно воспользоваться браузером и сформировать http-запрос в строке браузера или утилитой curl.

В учебной версии ограничено количество запускаемых серверов 1С. Поэтому рекомендуется закрыть конфигуратор и 1С Предприятие на этапе тестовых запросов. Результаты запросов можно будет увидеть в документе «Наряд» 1С Предприятии.

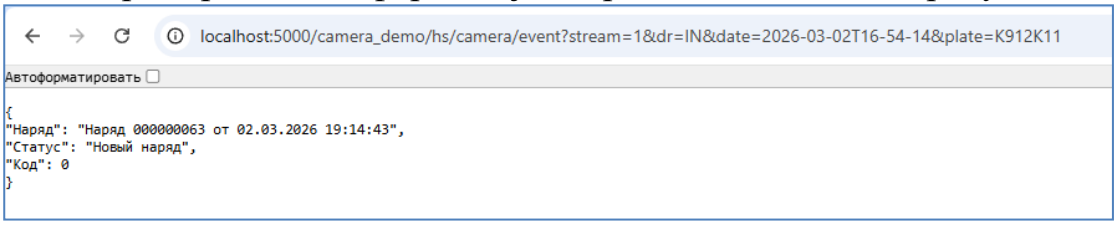
Пример строки запроса для учебной версии 1С и пользователя camera, у которого не задан пароль:

http://camera@localhost:5000/camera_demo/hs/camera/event?stream=1&dr=IN&date=2026-02-21T12:00:00&plate=K912K11

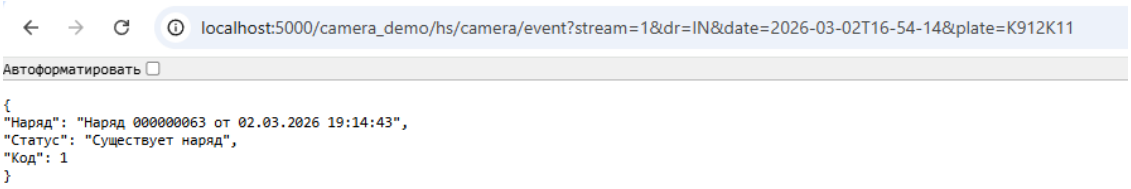
В случае наличии пароля, добавляем его в строку запроса:

http://camera:password@localhost:5000/camera_demo/hs/camera/event?stream=1&dr=IN&date=2026-02-21T12:00:00&plate= K912K11

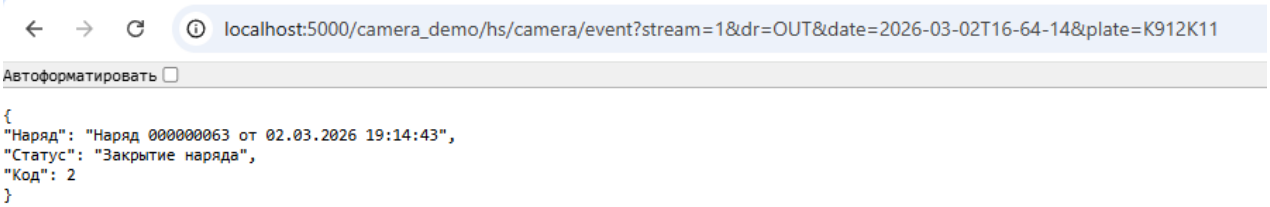
Пример ответа в формате json представлен ниже на рисунке:



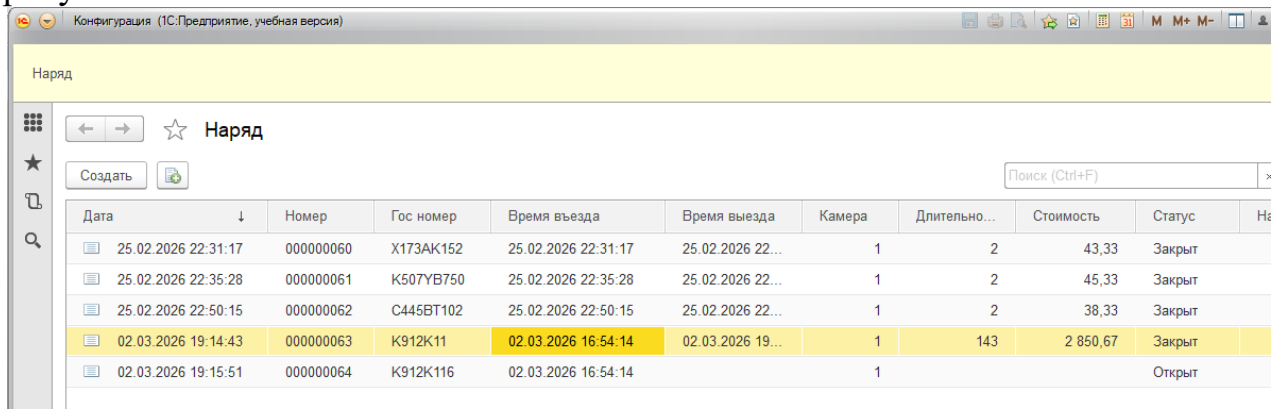
Вариант ответа при повторной отправке номера:



Тест ситуации закрытия наряда, при появлении того же номера с обратным направлением:



Результаты обработки http-запросов в документе «Наряд» 1С представлены на рисунке ниже.

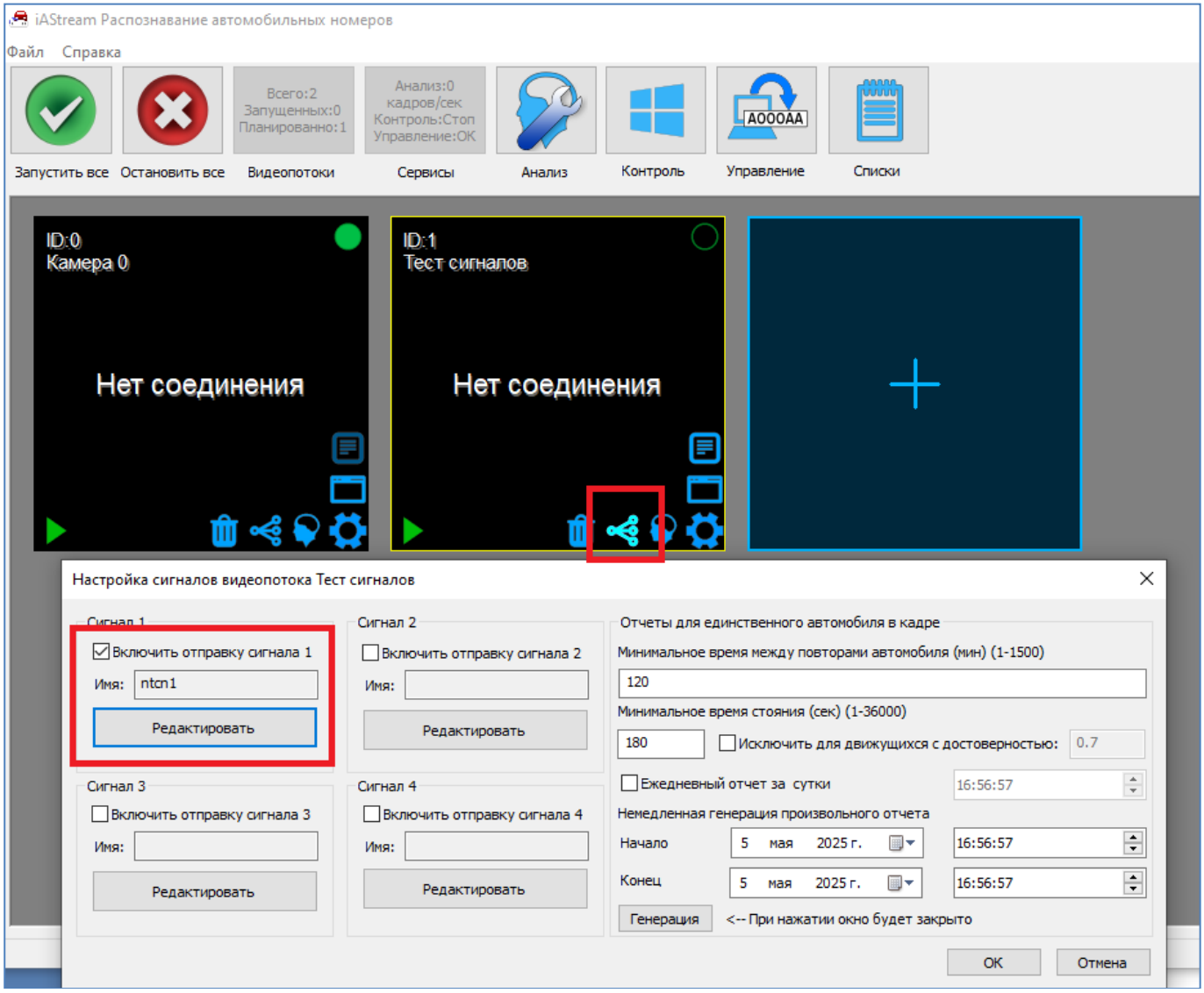


В окне отображается дата создания наряда, время въезда и выезда из http-запроса, длительность нахождения в мойке и тестовая стоимость услуги.

Результаты тестов показали работоспособность данного механизма интеграции iAStream с 1С.

Непосредственное тестирование

Первоначально настраиваем параметры отправки сигналов. Для этого в приложении iAStream выбираем требуемый видеопоток и нажимаем на значок редактирования параметров системы отправки сигналов о обнаружении событий.



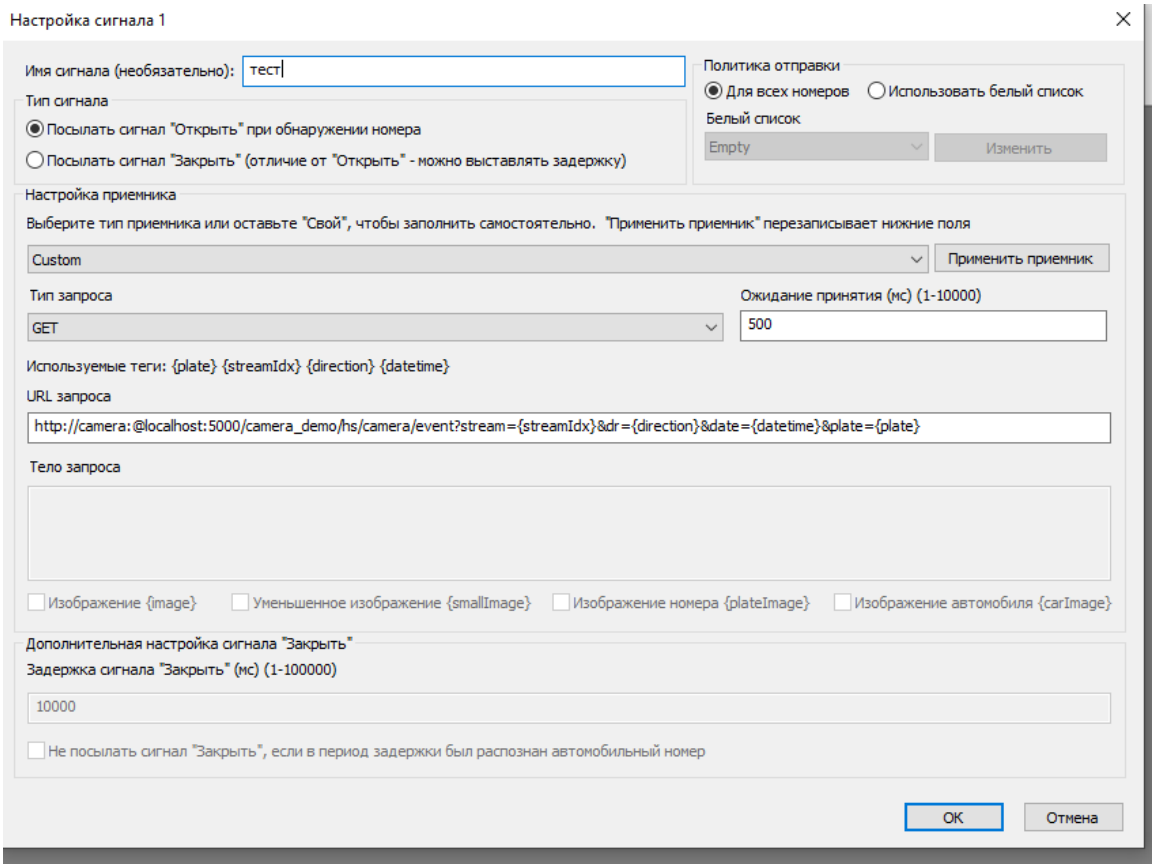
В данной работе рассматривается отправка сигнала 1: выставляем галочку «Включить отправку сигнала 1» и нажимаем кнопку «Редактировать» для задания параметров отправки сигнала.

Основные параметры отправки сигнала оставляем без изменения. Более подробно можно изучить возможности настройки отправки сигналов в документации iAStream.

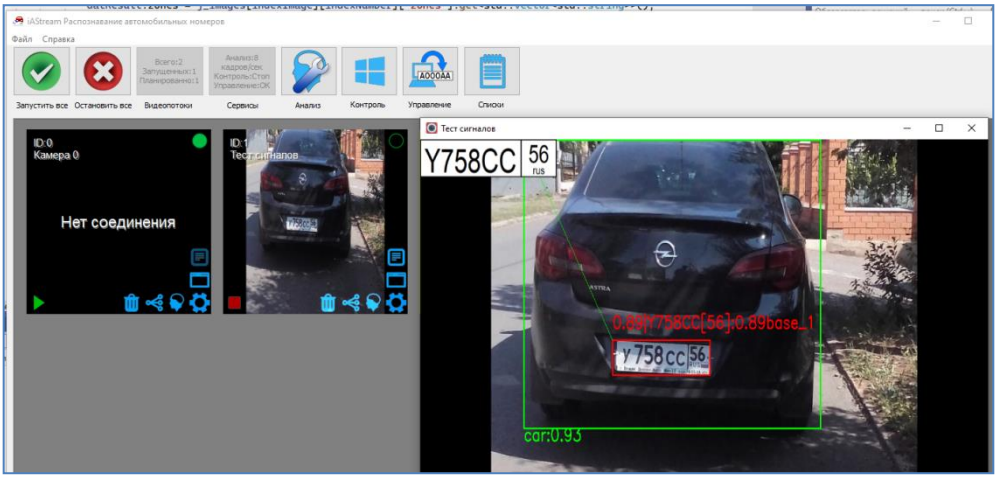
В рамках теста изменяем только раздел «Настройка приемника», который определяет параметры http-запроса. Изменяем следующие параметры:

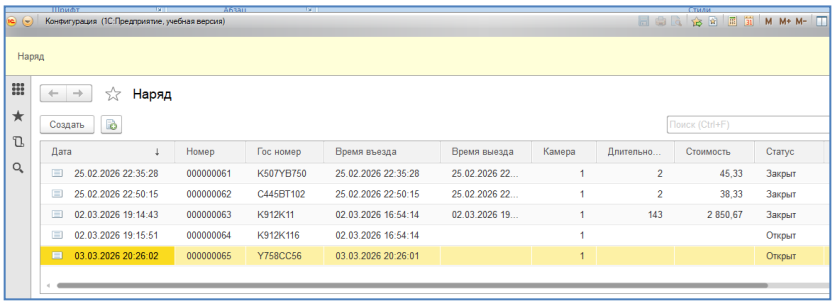
- тип приемника на значение «Custom»;
- тип запроса на «GET»;
- URL запроса:

«http://camera:@localhost:5000/camera_demo/hs/camera/event?stream={streamIdx}&dr={direction}&date={datetime}&plate={plate}».



После применяем наши настройки. При корректности всех настроек iAStream будет распознавать события и отправлять их на 1С. Пример окна iAStream на этапе работы представлен на рисунке ниже.





Результаты обработки сигналов должны зафиксироваться также в документе Наряд.

Таким образом, данная статья демонстрирует возможность автоматической передачи данных из программного комплекса iAStream в 1С. В частности, в данной

работе получается следующий результат: при проезде машины через камеру iAstream в 1С автоматически создается документ "Наряд", фиксируется время, а при выезде рассчитывается стоимость.

Приложение А
(справочное)
Модуль HTTP-сервиса

```
Функция ШаблонURLКамераПростойСигнал(Запрос)

	ЛоггированиеКамер.ЗаписатьВЛог("=== НАЧАЛО ОБРАБОТКИ ЗАПРОСА ===", "INFO");
	Параметры = Запрос.ПараметрыЗапроса;


  // Извлекаем нужные значения
  StreamIdx = Параметры.Получить("stream");
  Direction = Параметры.Получить("dr");
  DateTimeStr = Параметры.Получить("date");
  Plate = Параметры.Получить("plate");
	Результат = Новый Структура;
	//обработка данных
	Попытка
		StreamIdx =Число(StreamIdx);
	Исключение
		Результат.Вставить("Статус", "Номер камеры должен быть числом");
		Результат.Вставить("Код", -1);
		ЛоггированиеКамер.ЗаписатьВЛог("Ошибка: " + "Номер камеры должен быть числом" ,
"INFO");
		Возврат СоздатьОтветОшибки(400, Результат);
	КонецПопытки;
	ЛоггированиеКамер.ЗаписатьВЛог("Направление =" + Direction , "INFO");
	Если  Direction="OUT" Тогда
		 Direction = 1;
	 Иначе
		 Direction = 0;
	КонецЕсли;


  ЛоггированиеКамер.ЗаписатьВЛог("Дата =" + DateTimeStr , "INFO");
	ЛоггированиеКамер.ЗаписатьВЛог("Направление =" + Строка(Direction) , "INFO");
  // Проверка обязательных параметров
	Если StreamIdx = Неопределено ИЛИ Plate = Неопределено Тогда
	  Возврат СоздатьОтветОшибки(400,"Ошибка: не указаны stream или plate");
  КонецЕсли;


  // Преобразуем дату из строки
  ВремяСобытия = ТекущаяДата(); // по умолчанию
  Если DateTimeStr <> Неопределено Тогда
    Попытка
      ВремяСобытия = КамерыСервер.ПреобразоватьНестандартнуюДату(DateTimeStr);
    Исключение
      // Если не удалось преобразовать, оставляем текущую
			ЛоггированиеКамер.ЗаписатьВЛог("Ошибка" + ОписаниеОшибки() , "INFO");
    КонецПопытки;
  КонецЕсли;


		//ищем открытый наряд
	СуществующийНаряд = КамерыСервер.НайтиОткрытыйНаряд(Plate, StreamIdx);
		//если это новый наряд
	Результат = Новый Структура;
	Если СуществующийНаряд = Неопределено Тогда
		//создаем новый наряд
		НовыйНаряд  =  КамерыСервер.СоздатьНарядНаВъезд(Plate,  StreamIdx,  ВремяСобытия,
Direction);
		Результат.Вставить("Наряд", Строка(НовыйНаряд));
		Результат.Вставить("Статус", "Новый наряд");
		Результат.Вставить("Код", 0);
```

```
                ЛоггированиеКамер.ЗаписатьВЛог("Создан наряд " +  строка(НовыйНаряд), "INFO");
        Иначе
                //повторный сигнал = игнорируем
                Если СуществующийНаряд.НаправлениеВъезда = Direction Тогда
                        Результат.Вставить("Наряд", Строка(СуществующийНаряд));
                        Результат.Вставить("Статус", "Существует наряд");
                        Результат.Вставить("Код", 1);
                        ЛоггированиеКамер.ЗаписатьВЛог("Наряд существует " +
СуществующийНаряд, "INFO");
                Иначе
                        //р
                        НарядОбъект = СуществующийНаряд.ПолучитьОбъект();
                        РезультатРасчета =
РасчетСтоимости.РассчитатьСтоимость(НарядОбъект.ВремяВъезда,ВремяСобытия);
                        НарядОбъект.ВремяВыезда = ВремяСобытия;
                НарядОбъект.Длительность = РезультатРасчета.ДлительностьМинут;
                        НарядОбъект.Стоимость = РезультатРасчета.Стоимость;
                НарядОбъект.Статус = Перечисления.ТипСтатусаНаряда.Закрыт;
                НарядОбъект.Записать();
                        Результат.Вставить("Наряд", Строка(СуществующийНаряд));
                        Результат.Вставить("Статус", "Закрытие наряда");
                        Результат.Вставить("Код", 2);
                        ЛоггированиеКамер.ЗаписатьВЛог("Закрытие наряда " +
Строка(СуществующийНаряд), "INFO");
                КонецЕсли

        КонецЕсли;
        Возврат СоздатьОтветУспеха(Результат);
КонецФункции


Функция СоздатьОтветУспеха(Данные)

  Ответ = Новый HTTPСервисОтвет(200);

  // Преобразуем данные в JSON
  Если ТипЗнч(Данные) = Тип("Структура")
    Или ТипЗнч(Данные) = Тип("Соответствие")
    Или ТипЗнч(Данные) = Тип("Массив") Тогда

    //ЗаписьJSON = Новый ЗаписьJSON;
        СтрокаJSON = КамерыСервер.ПростаяЗаписьJSON(Данные);
                Ответ.Заголовки.Вставить("Content-Type", "application/json; charset=utf-8");
        Ответ.УстановитьТелоИзСтроки(СтрокаJSON, КодировкаТекста.UTF8);


  ИначеЕсли ТипЗнч(Данные) = Тип("Строка") Тогда
    // Если пришла строка, оборачиваем в JSON
    СтруктураОтвета = Новый Структура;
    СтруктураОтвета.Вставить("success", Истина);
    СтруктураОтвета.Вставить("message", Данные);
    СтрокаJSON = КамерыСервер.ПростаяЗаписьJSON(СтруктураОтвета);
                Ответ.Заголовки.Вставить("Content-Type", "application/json; charset=utf-8");
        Ответ.УстановитьТелоИзСтроки(СтрокаJSON, КодировкаТекста.UTF8);

  Иначе
    // Для других типов
                Ответ.Заголовки.Вставить("Content-Type", "application/json; charset=utf-8");
    Ответ.УстановитьТелоИзСтроки(Данные, КодировкаТекста.UTF8);
  КонецЕсли;

  Возврат Ответ;
КонецФункции

Функция СоздатьОтветОшибки(Код, Текст)
```

```
    Ответ = Новый HTTPСервисОтвет(Код);

    СтруктураОшибки = Новый Структура;
    СтруктураОшибки.Вставить("success", Ложь);
    СтруктураОшибки.Вставить("error", Истина);
    СтруктураОшибки.Вставить("status_code", Код);
    СтруктураОшибки.Вставить("message", Текст);
    СтруктураОшибки.Вставить("timestamp",  Формат(ТекущаяДата(), "ДФ=yyyy-MM-dd HH:mm:ss"));

    СтрокаJSON = КамерыСервер.ПростаяЗаписьJSON(СтруктураОшибки);
          Ответ.Заголовки.Вставить("Content-Type", "application/json; charset=utf-8");
    Ответ.УстановитьТелоИзСтроки(СтрокаJSON, КодировкаТекста.UTF8);

    Возврат Ответ;
КонецФункции
```

Приложение Б
(справочное)
Модуль журналирования событий программы

```
Процедура ЗаписатьВЛог(ТекстСообщения, Уровень = "INFO") Экспорт

  Попытка
    КаталогЛогов = "C:\temp\camera_logs\";

    // Пытаемся создать каталог (если существует - ошибки не будет)
    СоздатьКаталог(КаталогЛогов);

    // Формируем имя файла
    ИмяФайла = КаталогЛогов + "camera_" + Формат(ТекущаяДата(), "ДФ=yyyyMMdd") + ".log";

    // Формируем строку для записи
    ДатаСтрока = Формат(ТекущаяДата(), "ДФ=yyyy-MM-dd HH:mm:ss");
    СтрокаЗаписи = ДатаСтрока + " [" + Уровень + "] " + ТекстСообщения;

    // Записываем в файл
    ТекстовыйФайл = Новый ТекстовыйДокумент;

    ФайлЛога = Новый Файл(ИмяФайла);
    Если ФайлЛога.Существует() Тогда
      ТекстовыйФайл.Прочитать(ИмяФайла, "UTF-8");
    КонецЕсли;

    ТекстовыйФайл.ДобавитьСтроку(СтрокаЗаписи);
    ТекстовыйФайл.Записать(ИмяФайла, "UTF-8");

  Исключение
    // Полностью игнорируем ошибки логирования
  КонецПопытки;

КонецПроцедуры
```

Приложение В
(справочное)
Модуль КамерыСервер

```
Функция НайтиОткрытыйНаряд(ГосНомер, Камера) Экспорт

        ЛоггированиеКамер.ЗаписатьВЛог("Поиск открытого наряда для номера: " + ГосНомер + ", камера: " + Камера, "INFO");
        НомерКамеры = -1;
        Попытка
    НомерКамеры = Число(Камера); // "stream1" -> 0, "1" -> 1
  Исключение
    ЛоггированиеКамер.ЗаписатьВЛог("Не удалось преобразовать камеру в число: " + Камера, "ERROR");
  КонецПопытки;


  Запрос = Новый Запрос;
  Запрос.Текст =
  "ВЫБРАТЬ ПЕРВЫЕ 1
  |  Наряд.Ссылка,
  |  Наряд.ВремяВъезда
  |ИЗ
  |  Документ.Наряд КАК Наряд
  |ГДЕ
  |  Наряд.ГосНомер = &ГосНомер
  |  И Наряд.Камера = &Камера
```

```
    |  И Наряд.Статус = &Статус
    |УПОРЯДОЧИТЬ ПО
    |  Наряд.ВремяВъезда УБЫВ";

    Запрос.УстановитьПараметр("ГосНомер", ГосНомер);
    Запрос.УстановитьПараметр("Камера", НомерКамеры);
    Запрос.УстановитьПараметр("Статус", Перечисления.ТипСтатусаНаряда.Открыт);

          Попытка
      Результат = Запрос.Выполнить();

      Если Результат.Пустой() Тогда
        ЛоггированиеКамер.ЗаписатьВЛог("Открытых нарядов не найдено", "INFO");
        Возврат Неопределено;
      КонецЕсли;

      Выборка = Результат.Выбрать();
      Выборка.Следующий();

      ЛоггированиеКамер.ЗаписатьВЛог("Найден наряд: " + Выборка.Ссылка, "INFO");

      Возврат Выборка.Ссылка;
    Исключение
      ЛоггированиеКамер.ЗаписатьВЛог("Ошибка запроса: " + ОписаниеОшибки(), "ERROR");
      Возврат Неопределено;
    КонецПопытки;


КонецФункции

Функция СоздатьНарядНаВъезд(ГосНомер, Камера, ВремяСобытия, Direction) Экспорт

    НовыйНаряд = Документы.Наряд.СоздатьДокумент();
    НовыйНаряд.Дата = ТекущаяДата();
    НовыйНаряд.Камера = Камера;
    НовыйНаряд.ГосНомер = ГосНомер;
    НовыйНаряд.ВремяВъезда = ВремяСобытия;
    НовыйНаряд.Статус = Перечисления.ТипСтатусаНаряда.Открыт;
          НовыйНаряд.НаправлениеВъезда = Direction;
    НовыйНаряд.Записать();
          Возврат НовыйНаряд

КонецФункции

Функция ПростаяЗаписьJSON(Данные)   Экспорт

    ЗаписьJSON = Новый ЗаписьJSON;
    ЗаписьJSON.УстановитьСтроку();
    ЗаписатьJSON(ЗаписьJSON, Данные);
    Возврат ЗаписьJSON.Закрыть();
КонецФункции
Функция ПреобразоватьНестандартнуюДату(Знач СтрокаДаты) Экспорт

    ЛоггированиеКамер.ЗаписатьВЛог("Преобразование нестандартной даты: " + СтрокаДаты, "DEBUG");

    Если СтрокаДаты = Неопределено ИЛИ ПустаяСтрока(СтрокаДаты) Тогда
      Возврат ТекущаяДата();
    КонецЕсли;

    Попытка
      // Пример: "2026-02-25T15-54-14"

      // Разделяем дату и время по символу 'T'
      ПозицияТ = Найти(СтрокаДаты, "T");
      Если ПозицияТ = 0 Тогда
        // Если нет T, пробуем пробел
```

```
        ПозицияТ = Найти(СтрокаДаты, " ");
    КонецЕсли;

    Если ПозицияТ = 0 Тогда
        // Если нет разделителя, возвращаем текущую дату
        ЛоггированиеКамер.ЗаписатьВЛог("Не найден разделитель даты и времени", "WARNING");
        Возврат ТекущаяДата();
    КонецЕсли;

    // Извлекаем часть даты (до Т)
    ЧастьДаты = Лев(СтрокаДаты, ПозицияТ - 1);

    // Извлекаем часть времени (после Т)
    ЧастьВремени = Сред(СтрокаДаты, ПозицияТ + 1);

    // Разбираем дату: ГГГГ-ММ-ДД
    КомпонентыДаты = СтрРазделить(ЧастьДаты, "-", Ложь);
    Если КомпонентыДаты.Количество() < 3 Тогда
        Возврат ТекущаяДата();
    КонецЕсли;

    Год = Число(КомпонентыДаты[0]);
    Месяц = Число(КомпонентыДаты[1]);
    День = Число(КомпонентыДаты[2]);

    // Разбираем время: ЧЧ-ММ-СС
    КомпонентыВремени = СтрРазделить(ЧастьВремени, "-", Ложь);

    Часы = 0;
    Минуты = 0;
    Секунды = 0;

    Если КомпонентыВремени.Количество() >= 1 Тогда
        Часы = Число(КомпонентыВремени[0]);
    КонецЕсли;

    Если КомпонентыВремени.Количество() >= 2 Тогда
        Минуты = Число(КомпонентыВремени[1]);
    КонецЕсли;

    Если КомпонентыВремени.Количество() >= 3 Тогда
        Секунды = Число(КомпонентыВремени[2]);
    КонецЕсли;

    // Создаем дату
    Результат = Дата(Год, Месяц, День, Часы, Минуты, Секунды);

    ЛоггированиеКамер.ЗаписатьВЛог("Успешно преобразовано: " + Результат, "INFO");

    Возврат Результат;

  Исключение
    ЛоггированиеКамер.ЗаписатьВЛог("Ошибка преобразования даты: " + ОписаниеОшибки(),
"ERROR");
    Возврат ТекущаяДата();
  КонецПопытки;

КонецФункции
```