API
программного комплекса
ДинРобот-3

Евстигнеев Д.В.

07.12.2025

Оглавление

1. Введение

1.1. Назначение данного документа

В этом документе приводится описание API взаимодействия WEB-интерфейса экранного контента с программным комплексом ДинРобот-3.

Данный документ предназначается для экспертных разработчиков экранного контента робота под управлением программного комплекса ДинРобот-3.

1.2. Организация экранного WEB-контента в «ДинРобот-3»

Экранный контент робота в программном комплексе ДинРобот-3 реализуется на базе WEB-интерфейса. В роли WEB-сервера, с которого загружаются страницы WEB-контента, выступает сам программный комплекс «ДинРобот-3», в котором открыт Web-сервер экранного контента на порту 81 (или ином порту, заданном в «config.txt»). Базовым адресом сервера является «http://127.0.0.1:81» (81 – порт, про который шла речь ранее).

Экранный контент отображается на экране робота по средствам браузера Google Chrome, который запускается в режиме «--kiosk» (на полный экран без возможности возврата в оконный режим). Причем браузер Google Chrome запускается самим программный комплексом ДинРобот-3 с параметрами, позволяющими ему обходить запреты и предупреждения доступа к некоторым функциям браузера, требующими доступа, только по защищенному протоколу https. В силу того, что физически невозможно выписать SSL-сертификат на локальный сервер, браузер запускается с параметры, разрешающими ему работать с этими функциями без SSL-сертификата по незащищенному http-протоколу.

1.3. Безопасность экранного контента

Сервер экранного контента открывается исключительно на интерфейсе localhost (127.0.0.1), что не позволяет злоумышленникам получить доступ к этому серверу из вне.

В файле «C:\DynRobot3\config.txt», можно отключить это правило параметром LOCAL_HOST_ONLY=0 в секции [WEB_CONTROL_SERVER]. В этом случае сервер экранного контента будет открыт на всех сетевых интерфейсах робота. Однако, такое решение не безопасно, но дает возможность интеграции с другими устройствами.

1.4. Хранилище файлов экранного WEB-контента

Файлы экранного контента обычно хранятся в папке:

«C:\DynRobot3\WEBContent\<*название_контента*>\»

Точное название папки можно посмотреть в конфигурационном файле «C:\DynRobot3\config.txt» в секции [WEB_CONTENT_SERVER] параметр

«DIRECTORY». Разработчик экранного контента может создать свою папку экранного контента, указав к ней путь в параметре «DIRECTORY», например:

```
DIRECTORY = WEBContent/myContent/
```

Обратите внимание, в конце нужно указывать знак «/». Папка задается относительно папки «DynRobot3».

Можно также указать абсолютный путь к папке на диске, например: «C:\MyWebServer\content\».

1.5. Кодировка файлов

HTML-, JS-, CSS-файлы экранного контента передаются в кодировке UTF-8. Ответы сервера в подавляющем большинстве случаев формируются также в кодировке UTF-8. Исключением являются бинарные данные и те документы, которые специально запросили в иной кодировке.

Файлы .htaccess, через которые на серверах Apache задается кодировка файлов, не поддерживаются.

1.6. Методы http-запроса

Программный комплекс «ДинРобот-3» одинаково реагирует на запросы, сделанные как методом GET, так и методом POST в формате «application/x-www-form-urlencoded». Однако рекомендуется использовать метод POST, который в отличие от метода GET никогда не кэширует ответы сервера, что гарантирует получение от сервера актуального ответа.

Некоторые, специально оговоренные запросы необходимо передать в формате «multipart/form-data».

Для передачи запроса рекомендуется использовать технологию асинхронных запросов AJAX (XMLHttpRequest). Удачная (по мнению автора этого документа) реализация этой технологии реализована в функции «sendXHR(url, post, onResponse, onError)», реализованной в файле «std.js», приложенному к существующим контентам. Примеры в данном документе будут приводиться на основе этой функции. Если у разработчика контента есть свои предпочтения в реализации асинхронных запросов, то необходимо самостоятельно адаптировать пример под эти другие реализации.

1.7. Организация файлов и управляющий модулей WEB-сервера ДинРобот-3

Все файлы указанной рабочей папки WEB-контента транслируются в протокол HTTP на адрес «http://127.0.0.1:81/». Т.е. если в корневой папке экранного контента есть файл «a.html», то полный URL этого файла по протоколу HTTP будет «http://127.0.0.1:81/a.html». Относительный же URL-файла совпадает с его названием (с точностью до экранирования специальных символов в URL-адресе). Имена файлов, заданных русскими буквами, транслируется в кодировке UTF-8.

Соответственно также транслируются подпапки. Получить доступ к файлам, лежащим уровнем выше корневой папки WEB-контента обычными средствами не получится.

Управляющие модули программного комплекса ДинРобот-3 транслируются в то же URL-пространство адресов, что и обычные файлы. Модули имеют названия без расширения. Например, модуль «control» в адресное URL-пространство транслируется как:

http://127.0.0.1:81/control

Относительный URL этого модуля (относительно файлов контента) будет просто «control».

Файл index.html

Главным файлов контента рекомендуется сделать файл «index.html».

Файл «index2.html»

Файл «index2.html», который можно найти и в папке существующего контента и скопировать в свой контент, представляет собой систему распознавания речи, реализующую внизу экрана строку «Говорите:». При этом «index2.html» загружает файл «index.html» внутри созданного внутри элемента «iframe». Внешне это выглядит, как содержимое файла index.html, внизу которого имеется строка «Говорите:».

Файл «dynrobot.js»

К каждому html-файлу контента имеет смысл подключить файл скрипта **«dynrobot.js»** (который можно найти в папке существующего контента). Этот файл реализует связь программным комплексом «ДинРобот-3» через WebSocket, что позволяет оперативно передать в экранный контент команды от «ДинРобот-3», в т.ч. команды управления экранным контентом, формируемые через IScript3.

В IScript3 не будет работать функция SetBrowserURL, если страница контента не подключила файл «dynrobot.js» или не реализовала его функционал другими средствами.

Файл «std.js»

В файле «std.js» находится ряд функций стандартного функционала, используемого при разработке контентов. Например, там содержится функция sendXHR, производящая асинхронные AJAX-запросы. А также функция _html_, которая экранирует специальные символы в строке (например, заменяет «&» на «&») перед выводом в HTML.

Файл является не обязательным, Разработчик может использовать свои библиотеки вместо этого файла. Однако пример в данном документе будут приводиться на основе этого файла.

Файл «virtualkeyboard.js»

Файл подключает экранную виртуальную клавиатуру, реализованную на WEB-интерфейсе.

Файл является не обязательным, Разработчик может использовать свои библиотеки для вывода виртуальной клавиатуры.

Файл «contentui.js»

Файл является стандартной библиотекой для WEB-контентов. Внутри себя он подключает «dynrobot.js», «virtualkeyboard.js» и «std.js», а также «style.css», поэтому при использовании данной библиотеки достаточно к разрабатываемому html-файлу подключить только эту библиотеку.

Это является не обязательным. Разработчик может использовать собственные (привычные ему) средства разработки html-страниц.

Однако все примеры в этом документе будут приводиться с учетом подключения именно этого файла.

1.8. Формат формируемых сервером сообщений об ошибках

В случае ошибок сервер формирует сообщения об этих ошибках. Имеет смысл сделать общий обработчик этих ошибок прямо в функции, реализующей асинхронные http-запрос (AJAX), как это сделано в файле «std.js» (см. 1.6).

1. В случае, если запрашиваемый файл не найден формируется стандартная HTTP-ошибка 404.
2. В случае серьезных ошибок на сервере формируется HTTP-ошибка 500. Она же формируется при возникновении ошибок при запросе каких-либо сгенерированных сервером картинок (например, изображений с камер).
3. В большинстве случаев в случае ошибки формируется http-статус 200 OK, но текст ответа сервера имеет вид «**ERROR:** *текст ошибки в кодировке UTF-8*». Разработчик контента может проверять наличие слова «ERROR:» в самом начале ответа сервера, и расценивать ответ, как ошибку.

2. Модуль «control»

2.1. Запрос совместимости робота

Запрос аппаратных возможностей робота.

URL адрес http-запроса (без указания хоста):

```
control
```

Параметры запроса (метод POST или GET):

```
action=compatibles:get
```

Если ошибок нет, в ответ сервер формирует JSON:

```
{
  "reloadVersion":1,      // версия конфигурации. Обновляется при изменении hwconfig.txt
  "robotID":11231,        // ID робота. Если подключится к другому роботу, ID будет другой
```

```
  "wheels":"2WHEELS", // тип шасси: NONE, 2WHEELS, 3ROT, 3OMNI, 4OMNI, DRIVER
  "canStrafe":false,        // умеет ли ездить стрейфом
  "canTurn":false,          // имеет переключение режимов шасси на поворот на месте
  "map":"VISUAL",           // тип карты: NONE, VISUAL, DEPTH
  "localMap":false,         // имеется ли локальная карта местности
  "head":true,              // есть ли голова
  "lift":false,                   // имеется ли подъемник
  "canPark":true,                 // имеется ли автоматическая парковка на зарядку
  "speech":true,                  // имеется ли синтезатор речи
  "hasVolumeControl":true,  // имеется регулировка громкости
  "hasAudioIn":true,              // есть ли микрофон
  "languages":["RU","EN"],  // языки, на которых возможен синтез речи
  "motors":[{...},{....}],        // информация о моторах (здесь опущено)
  "cameras":                      // информация про камеры
  [
    // id; название; является ли камерой глубины
    {"id":1, "name":"Лицевая камера", "depth":false},
    {"id":2, "name":"Нижняя камера", "depth":false}
  ],
  "arm":                          // перечень манипуляторов (рук)
  [
    // id;  название;  если поддержка обратной кинематики
    {"id":1,"name":"Левая рука","canIK":false},
    {"id":2,"name":"Правая рука","canIK":false}
  ],
  "webContentURL":"http://127.0.0.1:81/index2.html",    // URL WEB-контента
  "chatbotFile":"IScript/main.chatbot",                 // файл чат-бота или пустая строка
}
```

Периодически сервер через главный WEBSocket (см. далее) может посылать сообщение «reload:state 2 11231» (Здесь: 2 – версия конфигурации; 11231 – идентификатор робота). Это означает, что конфигурация «налету» поменялась. JavaScript на странице может сравнить эту версию с последней запрошенной версии, и, если версия не совпадает, перезапросить конфигурацию.

Пример, вывести список камер робота на экран:

```
<html>
<body>
<script src="contentui.js"></script>

<div id=resultDiv></div>

<script>
sendXHR("control","action=compatibles:get",
    function(response)
    {
         var json = JSON.parse(response);
         for(var i=0; i < json.cameras.length; i++)
         {
            var div = document.createElement("div");
            div.innerHTML = _html_(json.cameras[i].name);
            resultDiv.appendChild(div);
         }
    },
```

```
    function(error)
    {
        errorBox(error);
    });
</script>
</body>
</html>
```

3. Видео с камер робота. Модули «video$_x$»

В ДинРобот-3 для каждой логической камеры с номерами 0, 1, 2, и т.д. существует соответствующий видеомодуль с названием video0, video1, video2 и т.д.

Любой запрос с URL-адреса этого модуля возвращается одна картинка с камеры в формате JPG.

При запросе картинки дополнительным параметром к запросу можно указать параметр «faces=1», что означает, что запрос изображения с камеры потребует также «пробуждения» системы распознавания лиц. В принципе, когда на роботе работают скрипты IScript3, то там и так, обычно, производится постоянное распознавание лиц. Но при выключенных скриптах (отжатой кнопки «авто») или, если скрипты не включили систему распознавания лиц, то система распознания лиц может оказаться выключенной. В то время, как разрабатываемый видеосервис (будь то «фото в обзразе», «предсказания», «робоаватар», «знакомство») требует, чтобы распознавание лиц было включено, для чего и служит данный параметр.

Также для борьбы с кэшом рекомендуется добавлять к каждому запросу картинки с камеры дополнительный случайный параметр, это не позволит браузеру закэшировать картинку, отправляемую видеомодулем (в ряде случаев директива «no-cache» в заголовке ответа браузером игнорируется).

URL адрес http-запроса (без указания хоста):

video0

video1

video2

…

Параметры запроса (метод POST или GET, опционально):

faces=1

Здесь: faces – опциональный параметр требующий «пробуждения» системы трекинга лиц.

Робот всегда организуется таким образом, что камера 0 для него – это основная камера для экранных фотосервисов.

Пример организации фотосервиса.

```
<html>
<body>

<img id=videoImage width=640 height=480>

<script>
CAMERA = 0; // номер камеры
videoImage.requestTimer = 0;  // декрементный таймер запроса

// функция вызывается при загрузке картинки. Взводит таймер запроса
// следующего фото на 40 мс
videoImage.onload = function() { this.requestTimer = 40; }

// метод запроса определим в самой картинке
videoImage.request = function()
{
      this.requestTimer = 1500; // таймаут запроса след. фото
      this.src='video'+CAMERA;
};

// устанавливаем основной таймер на частоту 20 мс,
// по которому будем заправить картинки
setInterval(function(dT)
      {
            videoImage.requestTimer-=dT;
            if (videoImage.requestTimer<=0) videoImage.request();
      },20, 20);
</script>
</body>
</html>
```

4. Модуль фотосервисов. Модуль «photoservice»

4.1. Сделать фото с камеры робота

Данный запрос заставляет робота сделать фотографию с указанной камеры и сохранить ее в указанный файл. Следует заметить, что сам запрос не возвращает само содержимое фотографии, а лишь управляет процессом съемки.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= photo

camera= номер камеры, обычно 0. Значение 0 используется по умолчанию, если этот параметр не указывать.

phtoFileName= название и путь к файлу (в формате JPG), куда будет сохранена фотография (кодировка имени файла – UTF-8). Если не указан полный путь, то используется путь, относительно папки экранного WEB-контента,

Например: «photo/231231123123.jpg». Скрипт фотосервиса робота должен сгенерировать уникальное название этого файла, например, с использованием функции системного времени.

maskFileName= (необязательный параметр), название и путь к файлу, где находится фоторамка, накладываемая на фото (кодировка имени файла – UTF-8). Если не указан полный путь, то используется путь, относительно папки экранного WEB-контента. Файл фоторамки должен быть в формате PNG-24. Если параметр не указан или пуст, а также если файл фоторамки не будет найден, то фото получается без рамки.

smallFileName= (необязательный параметр), название и путь к файлу, куда будет сохранена уменьшенная (облегченная) версия той же фотографии (кодировка имени файла – UTF-8). Например: «smallphoto/231231123123.jpg». Удобно, когда название большой и уменьшенной фотографии, совпадают, но лежат в разных папках. Такие уменьшенные фото используются для создания превью для фотогалереи. Если параметр не указан или пуст, а также если файл фоторамки не будет найден, то фото получается без рамки.

analyze: 1 – делать ли анализ фото на предмет рекомендаций (по умолчанию 0). Если анализ не требуется рекомендуется установить параметр в значение 0 или не указывайте, т.к. анализ требует от системы больше вычислительных ресурсов.

Если ошибок нет, запрос возвращает код рекомендации (число в строковом представлении):

- 0 – нет рекомендаций (анализ не производился).
- 1 – очень хорошее фото.
- 2 – очень темные лица на фото.
- 3 – нормальное фото. На фото один мужчина.
- 4 – нормальное фото. На фото один мужчина пола. Улыбается (детектор улыбки работает неважно).
- 5 – нормальное фото. На фото один мужчина. Грустный или злой.
- 6 – нормальное фото. На фото одна женщина.
- 7 – нормальное фото. На фото одна женщина. Улыбается (детектор улыбки работает неважно).
- 8 – нормальное фото. На фото одна женщина. Грустная или злая.
- 9 – на фото не обнаружено лиц.

- 10 – все лица смещены влево.
- 11 – все лица смещены вправо.
- 12 – все лица слишком высоко.
- 13 – все лица слишком низко.
- 14 – на фото группа из одних мужчин.
- 15 – на фото группа из одних женщин.
- 16 – на фото группа людей из мужчин и женщин.

Запрос может иметь длительный таймаут до 10 секунд. После получения результата на диске в указанном месте должна появится сама фотография, которую можно загрузить по ее URL.

4.2. Фото в образе

Запрос фото в образе позволяет запрашивать фото, в котором лицо человека вписано в какой-либо образом. В ответ на такой запрос сервер формирует JPG-изображение. При этом определяется пол человека и для него подставляет случайный образ. Виньетки для сервиса «фото в образе» общие для всех контентов и по умолчанию хранятся в папке «C:\DynRobot3\vignettes», где разбиты на подпапки «man» (мужские образы), и «woman» (женские).

Запрос JPG-изображений к данному сервису нужно производить постоянно по таймеру с перерывом 40 мс между моментами загрузки фото и запроса следующего. Но не следует запрашивать следующее фото, если не пришло предыдущее чаще, чем 1500 мс.

Управление сервисом происходит за счет передачи в запросе изображения управляющих параметров. Во избежание случаев сбоя в прохождение команды (например, в случае таймаута), параметры команды следует передать в каждом запросе, а сам запрос снабжается номером этого запроса. Web-сервер программного комплекса «ДинРобот-3» сравнивает номер из очередного запроса с номером предыдущего запроса и выполняет команду только в том случае, если номера не совпадают.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **vignette**

reqNumber= номер запроса. Рекомендуется инкрементировать при каждом новом запросе.

next= требование смены образа. нормальное значение 0 (не менять, по умолчанию). Значение 1 приводит к смене образа. Значение имеет смысл оставлять после этого постоянно до момента следующей команды, а изменить лишь reqNumber.

save= название и путь к файлу (в формате JPG), куда нужно сохранить картинку (кодировка имени файла – UTF-8). Необязательный параметр. Пустое значение (или отсутствие параметра) означает не сохранять. Значение имеет смысл оставлять после этого постоянно до момента следующей команды, а изменить лишь reqNumber.

small= название и путь к файлу (в формате JPG), куда нужно сохранить уменьшенную картинку (кодировка имени файла – UTF-8). Необязательный параметр. Пустое значение (или отсутствие параметра) означает не сохранять. Значение имеет смысл оставлять после этого постоянно до момента следующей команды, а изменить лишь reqNumber.

Если ошибок нет, то в ответ сервер отправляет JPG-картинку очередного фото в образе.

Пример:

```
<html>
<body>

<img id=photoImage>
<br>
<button onclick='doNext()'>Следующий образ</button>
<button id=saveBtn onclick='doSave()'>Сохранить</button>

<script>
isActive = true; // признак активности сервиса

// подготовка изображения
photoImage.requestTimer = 0; // таймер запроса
photoImage.reqNumber = 0;  // номер запроса
photoImage.next = 0;     // команда выбора сл.образа
photoImage.save='';        // команда на сохранение. куда сохранять
photoImage.small='';  // куда сохранять уменьш.копию
photoImage.stopService = false; // признак остановки сервиса
                        //после сохранения изображения


// на загрузку фото
photoImage.onload = function()
{
      // взвести таймер запроса сл.картинки на 40 мс
      this.requestTimer = 40;

      // остановить сервис, если команда сохранения удачно завершилась
      if (this.stopService) isActive = false;
}

// функция запроса изображения
photoImage.request = function()
{
      this.requestTimer=1500;
```

```
      // если есть требование на сохранение,
      // то установить признак остановки сервиса,
      // если следующая картинка успешно загрузится
      this.stopService = this.save!='';

      this.src='photoservice?'+
            '&action=vignettes'+
            '&reqNumber='+this.reqNumber+
            '&next='+this.next+
            '&save='+encodeURIComponent(this.save)+
            '&small='+encodeURIComponent(this.small);
}

// основной таймер
setInterval(function(dT)
      {
            if (isActive)
            {
                  // запросить фото по таймеру
                  photoImage.requestTimer-=dT;
                  if (photoImage.requestTimer <=0) photoImage.request();
            }
      }, 20, 20);

// Запрос следующего образа.
// Вместе с тем будем активизировать сервис, если он не активен
function doNext()
{
      photoImage.save = '';
      photoImage.small = '';
      photoImage.next = 1;
      photoImage.reqNumber++;

      // сделать сервис активным
      isActive = true;
      saveBtn.disabled = false;
}

// сохранить фото на следующем запросе
function doSave()
{
      saveBtn.disabled = true; // запретить нажимать кнопку повторно

      // краткое уникальное имя файла, куда сохранять
      var fileName = ((new DateTime()) - 0).toString() + ".jpg";

      photoImage.save = "photo/"+fileName;
      photoImage.small = "smallphoto/"+fileName;
      photoImage.next = 0;
      photoImage.reqNumber++;
}

</script>
</body>
</html>
```

4.3. Сервис «Робоаватар»

Сервис «робоаватар» позволяет сфоткать человека и сделать из его лица аватарку для робота. Используется только роботами, у которых экран вместо лица.

Сервис, лишь управляет процессом создания робоаватара, а не запрашивает фото для этого сервиса. Чтобы организовать предпросмотр для этого сервиса следует использовать сервис запроса видеоизображений с камеры 0.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action=**avatar**

Если ошибок нет, то в ответ сервер отправляет «OK».

4.4. Сервис знакомства

Сервис знакомства позволяет сформировать запрос на добавление новой персоны в базу данных лиц. Подразумевается, что человек к этому моменту стоит перед камерой робота.

Предварительно WEB-интерфейс страницы должен организовать процедуру ввода имени персоны. После чего необходимо запустить предпросмотр видео с камеры 0 обязательно с параметром faces=1, чтобы «разбудить» систему распознавания лиц. Через несколько секунд после ее запуска можно делать запрос сервиса знакомства.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **acq**

id= уникальный идентификатор добавляемой персоны. Строка в кодировке UTF-8. Под этим идентификатором будет создана подпапка в папке «FaceDB», куда будет записана фотография персоны.

name= имя персоны. Строка в формате UTF-8.

force= 0 или 1, признак принудительного добавления персоны (игнорировать схожесть персоны с одной из существующих персон в базе данных робота). Параметр дает возможность реализации сценария «это не я».

Если ошибок нет, то в ответ сервер отправляет:

- «OK» – персона успешно добавлена.

- JSON: {"id": "Дима123", "name": "Дима"} – такой человек уже есть базе данных, JSON указывает на соответствующую персону.
- «ERROR: *сообщение об ошибке*» – ошибка добавления персоны в базу данных.

Добавление персоны происходит не мгновенно, поэтому таймаут запроса должен быть не менее 5-6 секунд. Для добавления персоны требуется примерно 4 секунды (точное значение задано в «config.txt» параметром NEW_PERSON_PERIOD в секции [FACE_SYSTEM]). За это время выбирается кадр с лучшим ракурсом. На это время на экране для человека можно формировать превью с камеры 0 с обратным отсчетом.

4.5. Переименование персоны в базе данных лиц

Возможность переименования персоны в базе данных лиц дает возможность организовывать сценарии типа «Меня не так зовут».

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **renameperson**

id= идентификатор персоны. Строка в кодировке UTF-8. Идентификатор совпадает с названием подпапки персоны в папке «FaceDB».

name= новое имя персоны. Строка в формате UTF-8.

Если ошибок нет, то в ответ сервер отправляет «OK»:

4.6. Блокировка события «* hello»

Событие «* hello» отправляется в IScript3, как указание поздороваться с человеком. Но в некоторых случаях здороваться неуместно. Например, в момент проведения процедуры знакомства.

Поэтому у WEB-страницы есть возможность отправить запрос на то, чтобы отложить появление события «* hello» (и прочих событий, связанных с персоной) на 4 секунды. Такой запрос следует отправить на сервер каждые 2-3 секунды в течение того времени, когда событие «* hello» неуместно.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action=**lockHello**

Если ошибок нет, то в ответ сервер отправляет «OK».

4.7. Распознавание лица, пола и возраста. Сервис гадания

Запрос позволяет определить пол, возраст и, если это возможно, имя человек перед камерой робота.

Перед запросом требуется, чтобы работало видео с робота с параметром «faces=1», чтобы система распознавая лиц гарантировано работала.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action=**face**

Если ошибок нет, то в ответ сервер отправляет JSON:

```
{
  "age": 35,          // примерный возраст
  "gender": 0,        // пол: 0 – мужской, 1 – женский, -1 – не определен.
  "emotion":4,        // код эмоции:
                      // 0 – не изв. 1- злость, 2 – испуг, 3 – счастье, 4 - нейтрал,
                      // 5 – грусть, 6 – сюрприз. Определение работает неважно.
  "faceImageWidth":640,    // размер изображения, на котором детектировалось лицо
  "faceImageHeight:480,
  "faceRect":         // координаты прямоугольника лица на изображении
       {"left":100, "top":68, "right":211, "bottom":192},
   "person":          // персона. Может быть либо null, либо следующий объект:
       {"id": "Дима231232", "name": "Дима"}
}
```

4.8. Начать видеозапись. Сервис видеоотзыв

Запрос позволяет начать запись видео с камеры робота.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **videorecorder:start**

camera= (опционально) номер камеры или (-1) – камера по умолчанию, заданная в «config.txt» в качестве камеры для видеорекордера (параметр «camera» секция [VIDEO_RECORDER]). Если параметр не указан, то используется камера по умолчанию.

filename= название файла, куда будет производиться запись. Строка в формате UTF-8 Если полный путь не указан, то путь задается относительно папки WEB-контента. Параметр обязательный. Обычно имя файла генерируется по системному времени.

duration= (опционально) время записи в секундах, по умолчанию 60. Значение 0 или меньше запускает непрерывную запись серии видеофрагментов (лучше этим не пользоваться).

Если ошибок нет, то в ответ сервер отправляет «OK».

4.9. Завершить видеозапись. Сервис видеоотзыв

Запрос позволяет завершить запись видео с камеры робота.
URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **videorecorder:stop**

Если ошибок нет, то в ответ сервер отправляет «OK». Ответ «OK» будет получен, даже если видеозапись не велась.

4.10. Сброс сторожевого таймера видеозаписи. Сервис видеоотзыв

Чтобы подтвердить, что WEB-страница, запустившая видеозапись, всё еще активна и производит видеозапись, требуется не реже раза в три секунды (лучше каждую секунду) сбрасывать сторожевой таймер видеозаписи. Если это не сделать, то видеозапись завершится сама в ближайшие 3 секунды.
URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **videorecorder:kick**

Если ошибок нет, то в ответ сервер отправляет «OK». Ответ «OK» будет получен, даже если видеозапись не велась.

4.11. Сервис «Робо-художник»

4.11.1. Запуск сервиса «робо-художник»

Запрос запускает функцию «робо-художник». Эта функция ожидает очередного кадра видеоизображения, захватывает его, обрабатывает, и формирует изображение предварительного результата. Получить изображение можно с помощью запроса «action=roboart:getimage», а с помощью скрипта iScript3 (функция RoboArt()) можно получить массив линий для рисования роботом.
URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **roboart:start**

camera= 0 *// номер камеры, по умолчанию 0*

Если ошибок нет, то в ответ сервер отправляет «OK». Иначе сообщение об ошибке.

Подразумевается следующий сценарий:

1. Экранный контент робота реализует интерфейс предварительного просмотра видеоизображения с камеры.
2. Экранный контент ожидает нажатия определенной кнопки на экране. По ее нажатию запускается обратный отсчет, по истечению которого запрос кадров видеоизображения прекращается и формируется запрос:
   photoservice?action=roboart:start&camera=0
   Ожидается ответ «OK».
3. После получения ответа «OK», экранный контент запрашивает картинку с предварительным результатом по адресу:
   photoservice?action=roboart:getimage
   Полученная картинка отображается пользователю.
4. Экранный контент ожидает нажатия кнопки подтверждения результата. На кнопке подтверждения надпись, вроде: «Мне нравится, рисуй». Рядом кнопка «Еще раз», которая возвращает сценарий экранного контента к пункту 1.
5. По нажатию кнопки подтверждения экранный контент отправляет в iScript3 какое-нибудь событие, например, «* DRAW». Это можно сделать через функцию javascript-функцию sendToIScript из «dynrobot.js».
6. Скрипт на iScript3 ожидает событие «* DRAW». Обработчик события в iScript3 запрашивает массив координат линий для рисования роботом через функцию RoboArt() без параметров.
7. Далее скрипт управления роботом заставляет руку робота вывести на бумаге изображение, состоящее из линий.

4.11.2. Получение предварительной картинки сервиса «робо-художник»

Запрос получает картинку (jpeg) из сервиса «робо-художник». Предварительно должен быть выполнен запрос «action=roboart:start» и получен положительный результат запроса.

URL адрес http-запроса (без указания хоста):

photoservice?action=roboart:getimage

Если ошибок нет, то в ответ сервер отправляет изображение в формате JPEG.

4.11.3. Запрос уровня бинаризации сервиса «робо-художник»

Запрос получает уровень бинаризации изображения для сервиса «робо-художник».

URL адрес http-запроса (без указания хоста):

photoservice?action=roboart:getbinlevel

Если ошибок нет, то в ответ сервер отправляет число в текстовом представлении, текущего уровня бинаризации.

Уровень бинаризации должен быть в диапазоне 10..40. От него зависит качество рисуемой роботом картинки.

Уровень бинаризации можно изменить запросом «roboart:recalc».

4.11.4. Перестройка изображения сервиса «робо-художник»

Запрос перестраивает изображение сервиса «робо-художник» с другим уровнем бинаризации. Запрос следует отправлять после получения результатов запроса «roboart:start».

Получить изображение можно с помощью запроса «action=roboart:getimage», а с помощью скрипта iScript3 (функция RoboArt()) можно получить массив линий для рисования роботом.

URL адрес http-запроса (без указания хоста):

photoservice

Параметры запроса (метод POST или GET):

action= **roboart:recalc**

binlevel= 20 *// новый уровень бинаризации, по умолчанию 20*

Если ошибок нет, то в ответ сервер отправляет «OK». Иначе сообщение об ошибке.

Подразумевается следующий сценарий:

1. Экранный контент робота реализует интерфейс предварительного просмотра видеоизображения с камеры.
2. Экранный контент ожидает нажатия определенной кнопки на экране. По ее нажатию запускается обратный отсчет, по истечению которого запрос кадров видеоизображения прекращается и формируется запрос:
   photoservice?action=roboart:start&camera=0
   Ожидается ответ «OK»
3. После получения ответа «OK», экранный контент запрашивает картинку с предварительным результатом по адресу:
   photoservice?action=roboart:getimage
   Полученная картинка отображается пользователю.
4. Экранный контент отображает ползунок с настройкой уровня бинаризации. Текущее значение уровня бинаризации (оно же положение ползунка) запрашивается с помощью запроса:
   photoservice?action=roboart:getbinlevel

5. При изменении ползунка формируется запрос:
   photoservice?action=roboart:recalc
   в котором указывается новое значение уровня бинаризации. На этот запрос должен придти ответ «OK».
   Экранный контент, получив ответ «OK», запрашивает картинку предварительного просмотра с URL:
   photoservice?action=roboart:getimage
6. При этом экранный контент ожидается нажатия кнопки подтверждения результата. На кнопке подтверждения надпись, вроде: «Мне нравится, рисуй». Рядом кнопка «Еще раз», которая возвращает сценарий экранного контента к пункту 1.
7. По нажатию кнопки подтверждения экранный контент отправляет в iScript3 какое-нибудь событие, например, «* DRAW». Это можно сделать через функцию javascript-функцию sendToIScript из «dynrobot.js».
8. Скрипт на iScript3 ожидает событие «* DRAW». Обработчик события в iScript3 запрашивает массив координат линий для рисования роботом через функцию RoboArt() без параметров.
9. Далее скрипт управления роботом заставляет руку робота вывести на бумаге изображение, состоящее из линий.

5. Файловая операционная система. Модуль «os»

Модуль предназначен для реализации взаимодействия с файлами и папками на диске робота. Например, с его помощью можно получить список файлов в фотогалереи робота, а также загрузить файлы, лежащие в системных папках робота, например, взять фотографию человека из папки FaceDB, а также создавать или читать текстовые файлы на диске робота.

5.1. Получить содержимое файла с диска робота

Читает файл из любой папки с диска робота и возвращает его содержимое.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=get&file=*имяФайла*&content-type=*типКонтента*

Здесь:

имяФайла – название файла (кодировка UTF-8). Если не указан полный путь к файлу (например: «C:/MyPhoto/info.txt»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/Дима1231231231/image0.jpg»

типКонтента – необязательный параметр. Указывает, какое значение параметра «Content-Type» будет в заголовке http-ответа сервера. Тем самым, можно менять кодировки текстовых файлов. Например, указав: «plain/text; encoding=windows-1251». По умолчанию данный параметр принимает значение: «application/octet-stream».

Если ошибок нет, в ответ сервер отправляет содержимое указанного файла. При этом в заголовке HTTP-ответа будет указан требуемое значение «Content-Type».

Пример, вывести содержимое файла «config.txt» на экран, при условии, что рабочая папка контента находится в «C:\DynRobot3\WEBContent\test», и учитывая, что этот файл использует кодировку «windows-1251»:

```
<html>
<body>
<script src="contentui.js"></script>

<textarea id=editorTextarea style='width:100%' rows=10></textarea>

<script>
var post = "action=get";
    post+="file="+encodeURIComponent("../../config.txt");
    post+="content-type="+
            encodeURIComponent("text/plain; encoding=windows-1251");
sendXHR("os",post,
    function(response)
    {
         editorTextarea.value = response;
    },
    function(error)
    {
        errorBox(error);
    });
</script>
</body>
</html>
```

5.2. Передача файла на диск робота (upload)

Команда предназначена для реализации процедуры передачи файлов на сервер, что больше подходит для реализации управляющего контента.

Команда должна быть реализован по средствам метода POST в формате «multipart/form-data», в противном случае метод выдает ошибку.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST, формат: «multipart/form-data»):

action: put

file: содержимое файла и его название (HTML формирует такое поле автоматически при использовании на странице <input type=file>.

path: путь к файлу (кодировка UTF-8). Если не указан полный путь, то используется путь, относительно папки экранного WEB-контента, однако, относительно этой папки можно указывать папки, находящиеся на один или несколько уровней выше, например: «../../mycontent».

backup: 1 – делать ли резервную копию старого файла. Резервная копия будет делаться только за счет присутствия данного параметра в запросе независимо от его содержимого. Если резервную копию делать не нужно, параметра не должно быть в самом запросе. При создании резервной копии к названию старого файла добавляется расширение «.bak». Например, файл с названием «a.txt» будет переименован в «a.txt.bak»

Отметим, что, если робот работает под Windows, то при попытке заменить «DynRobot3.exe», автоматически вместо этого создается файл «DynRobot3.exe_».

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

– «ERROR: Невозможно создать файл!» (не удалось даже открыть файл на запись, что-то не то с названием файла или нет доступа к указанной папке).
– «ERROR: Невозможно записать данные в файл!» (не удалось записать все содержимое файла, возможно закончилось место на диске).

5.3. Запись файла на диск робота (write)

В отличие от команды «put», данная команда позволяет сформировать тело файла с помощью javascript и записать его на диск робота.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action= write

file= название файла (кодировка названия файла – UTF-8). Если не указан полный путь, то используется путь, относительно папки экранного WEB-контента, однако, относительно этой папки можно указывать папки,

находящиеся на один или несколько уровней выше, например: «../../mycontent/a.txt».

data= содержимое файла.

encoding= кодировка файла. Поддерживается только значение «Windows-1251» или «UTF-8». При этом, если в текст строки данного параметра входит подстрока «UTF-8» или «utf-8», то данные остаются в неизменном виде, как их сформировал javascript (javascript всегда передает данные в кодировке UTF-8). В противном случае данные переводятся в кодировку windows-1251.

backup= 1 – делать ли резервную копию старого файла. Резервная копия будет делаться только за счет присутствия данного параметра в запросе независимо от его содержимого. Если резервную копию делать не нужно, параметра не должно быть в самом запросе. При создании резервной копии к названию старого файла добавляется расширение «.bak». Например, файл с названием «a.txt» будет переименован в «a.txt.bak»

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

- «ERROR: Невозможно создать файл!» (не удалось даже открыть файл на запись, что-то не то с названием файла или нет доступа к указанной папке).
- «ERROR: Невозможно записать данные в файл!» (не удалось записать все содержимое файла, возможно закончилось место на диске).

Пример, создать на диске файл «a.txt» и записать туда «Привет, мир!» в кодировке Windows-1251.

```
<html>
<body>
<script src="contentui.js"></script>

<button onclick='createMyFile()'>Нажми меня</button>

<script>
function createMyFile()
{
   var fileName = "a.txt";
   var text = "Привет, мир!";
   var encoding = "Windows-1251";

   var post = "action=write"+
        "file="+encodeURIComponent(fileName)+
        "data="+encodeURIComponent(text)+
        "encoding="+encodeURIComponent(encoding);
   sendXHR("os", post, null,
      function(error)
      {
```

```
        errorBox(error);
      });
}
</script>
</body>
</html>
```

5.4. Дописать в конец файла на диск робота (append)

Команда дописывает данные в конец файла, а если его не существует, то создает его. Команда не создает резервной копии файла. В остальном команда аналогична команде write.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action= append

file= название файла (кодировка названия файла – UTF-8). Если не указан полный путь, то используется путь, относительно папки экранного WEB-контента, однако, относительно этой папки можно указывать папки, находящиеся на один или несколько уровней выше, например: «../../mycontent/a.txt».

data= содержимое файла.

encoding= кодировка файла. Поддерживается только значение «Windows-1251» или «UTF-8». При этом, если в текст строки данного параметра входит подстрока «UTF-8» или «utf-8», то данные остаются в неизменном виде, как их сформировал javascript (javascript всегда передает данные в кодировке UTF-8). В противном случае данные переводятся в кодировку windows-1251.

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

- «ERROR: Невозможно открыть или создать файл!» (не удалось даже открыть файл на запись, что-то не то с названием файла или нет доступа к указанной папке).
- «ERROR: Невозможно записать данные в файл!» (не удалось записать все содержимое файла, возможно закончилось место на диске).

5.5. Удаление файлов с диска робота

Запрос удаляет файлы из любой папки с диска робота.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=del&files=*имена Файлов*

Здесь:

имена Файлов – один или несколько названий файлов (кодировка UTF-8), перечисленных через знак перевода строки «\n» (0x0A). Если не указан полный путь к файлу (например: «C:/MyPhoto/info.txt»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/Дима1231231231/image0.jpg»

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

- «ERROR: Не все файлы были успешно удалены!»;
- «ERROR: Нечего удалять!»

Например, удалить файл «a.txt» и «b.txt» в папке, на уровень выше папки контента:

```
<html>
<body>
<script src="contentui.js"></script>

<button onclick='delMyFiles()'>Удалить файлы</button>

<script>
function delMyFiles()
{
   var files = ["../a.txt", "../b.txt"];

   var post = "action=del"+
        "files="+encodeURIComponent(files.join("\n"));

   sendXHR("os", post, null,
      function(error)
      {
        errorBox(error);
      });
}
</script>
</body>
</html>
```

5.6. Создание директории на диске робота

Запрос создает директорию на диске робота.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=mkdir&file=*название Директории*

Здесь:

названиеДиректории –название создаваемой директории (кодировка UTF-8). Если не указан полный путь к файлу (например: «C:/MyPhoto»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/person1»

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

– «ERROR: Не удалось создать директорию!»

5.7. Удаление директорий с диска робота

Запрос удаляет указанные директории из любой папки с диска робота.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=rmdir&files=*именаДиректорий*

Здесь:

именаДиректорий – один или несколько названий директорий, перечисленных через знак перевода строки «\n» (0x0A). Кодировка UTF-8. Если не указан полный путь к файлу (например: «C:/MyPhoto/info.txt»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/Дима123».

Директории удаляются вместе с их содержимым.

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

– «ERROR: Не все папки были успешно удалены!»;

– «ERROR: Нечего удалять!»

5.8. Переименование файла или директории на диске робота

Запрос переименовывает файл или директорию на диске робота, включая перенос файла в другое место диска.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=rename&oldFileName=*староеИмя*&newFileName=*новоеИмя*

Здесь:

староеИмя –Название существующего файла или папки (кодировка UTF-8). Если не указан полный путь к файлу (например:

«C:/MyPhoto/info.txt»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/info.txt».

новоеИмя – Новое название файла или папки (кодировка UTF-8). Если не указан полный путь к файлу (например: «C:/MyPhoto/info.txt»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/info2.txt». Заметим, новое название файла также должно содержать путь, иначе файл будет перенесен в папку WEB-контента.

Если ошибок нет, в ответ сервер отправляет «OK».

Могут быть следующие сообщения об ошибках (кроме стандартных):

– «ERROR: Не удалось переименовать файл или директорию!».

5.9. Проверка существования файла или директории

Запрос проверяет существование указанного файла или директории на диске робота.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=exists&file=*имяФайла*

Здесь:

имяФайла – Название проверяемого файла или папки (кодировка UTF-8). Если не указан полный путь к файлу (например: «C:/MyPhoto/info.txt»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB/info.txt».

Если ошибок нет, в ответ сервер отправляет ответ:

«1» – файл или директория существует;

«0» – файл или директория не существует.

5.10. Запрос оглавление директории на диске робота

Запрос получает перечень файлов и поддиректорий в указанной директории на роботе.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=dir&path=*путь*

Здесь:

путь – Путь к папке (кодировка UTF-8). Если не указан полный путь к файлу (например: «C:/MyPhoto/»), то путь к файлу задается относительно папки экранного контента робота, однако допускается указывать путь, используя знаки «../» (на уровень выше). Например: «../../FaceDB». Наличие знака слеша в конце не обязательное.

Если ошибок нет, в ответ сервер отправляет JSON:

```
[  {
     "fileName":".",       // имя файла или папки
     "isDir":true,         // признак, что это директория
   },
   {
     "fileName":"..",      // имя файла или папки
     "isDir":true,         // признак, что это директория
   },
   {
      "fileName":"myfile1.txt",      // имя файла или папки
      "isDir":false,         // признак, что это директория
   },
   {
      "fileName":"myfile2.txt",      // имя файла или папки
      "isDir":false,         // признак, что это директория
   },
  … ]
```

Важно, что в начале списка идут папки, а потом файлы. Кодировка имен файлов, заданных русскими буквами – UTF-8.

Например, вывести на экран все JPG-картинки из подпапки «gallery» WEB-контента:

```
<html>
<body>
<script src="contentui.js"></script>
<div id=galleryDiv></div>

<script>
   var path = "gallery/";
   var post = "action=dir&path=" + encodeURIComponent(path);
   sendXHR("os", post,
      function(response)
      {
         var json = JSON.parse(response);
         for(var i=0; i < json.length; i++)
         {
              if (json[i].fileName.match(/\.jpg$/))
              {
                 var img = new Image();
                 img.style.margin="10px 10px 10px 10px";
                 img.src = path + json[i].fileName;
                 galleryDiv.appendChild(img);
              }
```

```
            }
        },
        function(error)
        {
          errorBox(error);
        });
</script>
</body>
</html>
```

5.11. Запрос перечня дисков на роботе

Запрос получает перечень дисков на роботе (под Linux возвращает только один диск «\\ - Корень»).

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=disks

Если ошибок нет, в ответ сервер отправляет JSON с массивом строк:

["C:\\ - Жесткий", "D:\\ - Сменный", "G:\\ - CD-ROM", "Y:\\ - Сетевой", "Z:\\ - RAM-диск"]

5.12. Перезапуск «ДинРобот-3»

Запрос перезапускает ДинРобот-3. Обычно требуется для завершения обновления исполнимого файла программы.

URL адрес http-запроса (без указания хоста):

os

Параметры запроса (метод POST или GET):

action=restart

Если ошибок нет, в ответ сервер отправляет «OK».

6. Модуль управления жестами «gestures»

6.1. Запрос списка манипуляторов и жестов

Запрос получает список манипуляторов (рук) и их жестов.

URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action=list

Если ошибок нет, в ответ сервер отправляет следующий JSON:

```
[
 {
```

```
    "armName": "Левая рука",       // название манипулятора
    "armId": 0,                    // id руки
    "version": 11,                 // версия файла жестов, увеличивается при изменениях
    "canIK": true,                 // поддерживает ли обратную кинематику
    "gestures":                    // список жестов
    [
       {
         "name": “Жест0",          // название жеста
          "world": false,          // жест в мировой СК
          "showOnScreen": 1    // отображать жекст на главном экране управления
       },
       {
         "name": “Жест1",          // название жеста
          "world": false,          // жест в мировой СК
          "showOnScreen": 0    // отображать жекст на главном экране управления
       },
       …
    ]
  },
 {
    "armName": "Правая рука",      // название манипулятора
    "armId": 1,                    // id руки
    "version": 11,                 // версия файла жестов, увеличивается при изменениях
    "canIK": true,                 // поддерживает ли обратную кинематику
    "gestures":                    // список жестов
    [
       {
         "name": “Жест0",          // название жеста
          "world": false,          // жест в мировой СК
          "showOnScreen": 1        // отображать жекст на главном экране управления
       },
       {
         "name": “Жест1",          // название жеста
          "world": false,          // жест в мировой СК
          "showOnScreen": 0        // отображать жекст на главном экране управления
       },
       …
    ]
  }
]
```

Например, вывести список рук у робота:

```
<html>
<body>
<script src="contentui.js"></script>
<div id=resultDiv></div>
<script>
   sendXHR("gestures", "action=list",
      function(response)
      {
       var json = JSON.parse(response);
       for(var i=0; i < json.length; i++)
       {
          var div = resultDiv.appendChild(document.createElement("div"));
```

```
            div.innerHTML = _html_(json[i].armName);
         }
        },
        function(error)
        {
          errorBox(error);
        });
</script>
</body>
</html>
```

6.2. Запрос списка жестов главного экрана

Запрос получает список тех жестов, которые отображаются на главном экране.

URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action=scrlist

Если ошибок нет, в ответ сервер отправляет следующий JSON:

```
{
  "version": 11,          // версия файла жестов, увеличивается при изменениях
  "gestures":             // список жестов
  [ “Жест0", "Руки вверх", "Дай руку", "Обнимашки1", "Обнимашки2"]
}
```

6.3. Переименование жеста

Запрос позволяет переименовать жест.

URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action= rename

arm= id руки, по умолчанию 0

name= старое название жеста.

newname= новое название жеста.

showOnScreen= показывать ли на главном экране: 0 – не показывать, 1 – показывать, (-1) – не изменять.

Если ошибок нет, то в ответ сервер отправляет число в текстовом представлении, соответствующее версии нового файла жестов.

6.4. Удаление жеста

Запрос позволяет удалить жест.

URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action= delete

arm= id руки, по умолчанию (-1) – для всех рук.

name= название жеста.

Если ошибок нет, то в ответ сервер отправляет число в текстовом представлении, соответствующее версии нового файла жестов.

6.5. Добавление жеста

Запрос позволяет добавить текущее положение руки в качестве нового жеста. Если жест существует, то выдает ошибку.

URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action= add

arm= id руки, по умолчанию (-1), для всех рук.

name= название жеста.

world= запомнить жест в мировых координатах (0 или 1). По умолчанию 0.

showOnScreen= отображать жест на главном экране (0 или 1).

Если ошибок нет, то в ответ сервер отправляет число в текстовом представлении, соответствующее версии нового файла жестов.

6.6. Обновление положения жеста

Запрос позволяет обновить координаты жеста. При этом координаты берутся из текущее положения руки. Обновляемый жест должен существовать.

URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action= update

arm= id руки, умолчанию (-1) – для всех рук.

name= название жеста.

world= запомнить жест в мировых координатах (0 или 1). По умолчанию 0.

Если ошибок нет, то в ответ сервер отправляет число в текстовом представлении, соответствующее версии нового файла жестов.

6.7. Запуск жеста

Запрос позволяет запустить жест.
URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action= run

arm= id руки или -1 (на всех руках). По умолчанию -1.

name= название жеста.

Если жест был задан в мировых координатах, то при движении используется линейная интерполяция в декартовой системе координат. Иначе жест выполняет интерполяцию в обобщенной системе координат.

Если ошибок нет, то в ответ сервер отправляет «OK».

6.8. Удаление всех жестов

Запрос позволяет удалить все жесты со всех рук
URL адрес http-запроса (без указания хоста):

gestures

Параметры запроса (метод POST или GET):

action=deleteall

Если ошибок нет, то в ответ сервер отправляет «OK».

7. Модуль печати фото

7.1. Отправка фото на печать

В зависимости от параметра «LOCAL_PRINTER» секции [PRINTER] в файле «hwconfig.txt», печать фото может производиться либо на внешнем удаленном принтере через утилиту printclient.exe, которую можно скачать с самого робота (ссылка есть на странице авторизации), либо на встроенном в робота принтере (если он есть).

В случае печати на удаленном принтере все фотографии ставятся в очередь на печать, которая сохраняется даже после выключения робота в файле «printque.txt».

URL адрес http-запроса (без указания хоста):

print

Параметры запроса (метод POST или GET):

file=*имяФайла*

Где: *имяФайла* – название файла для печати (UTF-8). Если не задан полный путь, то путь к файлу задается относительно папки контента.

Если ошибок нет, то в ответ сервер отправляет «OK».

Если локальный принтер был занят, сервер пришлет ошибку:
«ERROR: Принтер занят!»

8. Модуль SMTP. Сервис отправки почты

8.1. Запрос отправки сообщения на email

Запрос позволяет отправить письмо на email. Имя и email отправителя задаются в «config.txt» в секции [SMTP]. Письмо забирается в очередь на отправку мгновенно, а сервис отправляет письма асинхронно в фоновом режиме при наличии связи с Интернет.

URL адрес http-запроса (без указания хоста):

smtp

Параметры запроса (метод POST или GET):

to= email адресата.

subject= тема письма (UTF-8).

text= тело письма. Если текст начинается с «<html>», то считается, что всё письмо в формате HTML (Content-Type: text/html) иначе считается, что письмо в обычном текстовом формате (Content-Type: text/plain).

attachFile= (опционально) название прикрепляемого файла (кодировка названия – UTF-8) или пустая строка, если нет прикрепленного файла. Если не задан абсолютный путь, то путь задается относительно папки WEB-контента.

attachMime= (опционально) MIME файла, например, «image/jpeg».

attachFile2= (опционально) название прикрепляемого файла 2 (кодировка названия – UTF-8) или пустая строка, если нет прикрепленного файла. Если не задан абсолютный путь, то путь задается относительно папки WEB-контента.

attachMime2= (опционально) MIME файла 2, например, «image/jpeg».

logFile= (опционально) название файла лога (кодировка названия UTF-8) или пустая строка, если нет файла лога. Если не задан абсолютный путь, то путь задается относительно папки WEB-контента. В этот файл система допишет новой строкой e-mail и название первого прикрепляемого файла через знак табуляции. Хоть это текстовый файл, придайте этому файлу расширение «xls». Это позволит открывать данный файл с помощью Excel, который

автоматически определяет формат файла и открывает его, разделив на столбцы по знаку табуляции и строки по знаку переноса строки.

Если ошибок нет, то в ответ сервер отправляет «OK».

9. Главный WebSocket

9.1. О главном WebSocket

В программном комплексе «ДинРобот-3» для непрерывной двухсторонней связи браузера и программного комплекса существует механизм главного WebSocket. Страницы Web-контента должны подключиться к главному WebSocket по адресу «**ws://127.0.0.1:81/main**» (порт 81 – это порт экранного контента), и ждать команд от «ДинРобот-3». Некоторые команды от браузера в «ДинРобот-3» также отправляются через главный WebSocket.

Страница Web-контента должна отслеживать разрыв связи по главному WebSocket, и в случае разрыва восстанавливать связь.

При использовании файла «dynrobot.js» выше описанный функционал реализован автоматически. При этом на ряд основных команд «dynrobot.js» реагирует самостоятельно, поэтому разработчику контента достаточно подключить этот файл к Web-странице контента. Напомним, что подключение файла «contentui.js» автоматически подключает «dynrobot.js» в Web-странице. Поэтому к HTML-файлу страницы достаточно подключить «contentui.js».

9.2. Функция «playSpeech» из «dynrobot.js»

JavaScript-функция «playSpeech», реализованная в «dynorobt.js», воспроизводит речь через программный комплекс «ДинРобот-3».

```
playSpeech(text);      // сказать одну фразу.
playSpeech([text1, text2, text3, text4…]);   // сказать одну из фраз,
                              // выбрав ее из массива случайным образом.
```

Например:

```
<html>
<body>
<script src="contentui.js"></script>
<script>
playSpeech("Это главная страница");
</script>
</body>
</html>
```

При этом для синтеза речи используется язык, указанный в программном комплексе, как язык по умолчанию. Если ранее использовался другой язык, то производится переключение на язык по умолчанию.

Сама функция playSpeech отправляет через главный WebSocket в программный комплекс «ДинРобот-3» текстовое сообщение «say *текст*», где «*текст*» – это текст, который требуется произнести. Текст должен быть в кодировке UTF-8 без экранирования каких-либо символов, но все знаки переноса строки должны быть из него удалены или заменены на пробел.

9.3. Функция «playSpeechLang» из «dynrobot.js»

JavaScript-функция «playSpeechLang», реализованная в «dynorobt.js», воспроизводит речь через программный комплекс «ДинРобот-3» указанным языком. Для указания языка используется глобальная переменная speech_language. Например: speech_language="RU".

```
speech_language="EN";      // указать английский
playSpeechLang(text);      // сказать одну фразу.
speech_language="RU";      // указать русский
playSpeechLang([text1, text2, text3, text4…]);   // сказать одну из фраз,
                           // выбрав ее из массива случайным образом.
```

Сама функция playSpeechLang отправляет через главный WebSocket в программный комплекс «ДинРобот-3» текстовое сообщение «say2 *lang текст*», где: «*lang*» – язык, указанный в переменной speech_language; «*текст*» – текст, который требуется произнести. Текст должен быть в кодировке UTF-8 без экранирования каких-либо символов, но все знаки переноса строки должны быть из него удалены или заменены на пробел.

9.4. Функция «sendToIScript» из «dynrobot.js»

JavaScript-функция «sendToIScript» отправляет в программный комплекс «ДинРобот-3» команду в IScript3. Команда должна начинаться со знаков «* ». Например: «* PRESENTMODE».

В IScript3 команда попадает как событие, отслеживаемое с помощью фреймовой структуры базы знаний (см. фреймы в руководстве по IScript3).

Разработчик контента может самостоятельно придумать свой набор команд для взаимодействия Web-контента со скриптом IScript3, если ему это необходимо.

```
sendToIScript(command);     // отправить команду в IScript3
```

Например:

```
<html>
<body>
<script src="contentui.js"></script>
<script>
sendToIScript("* SPEECH OFF");
</script>
</body>
</html>
```

Сама же функция sendToIScript отправляет команду в «ДинРобот-3» через главный WebSocket с помощью текстового сообщения «iscript:put *команда*», где «*команда*» – строка команды, кодировка UTF-8, в одну строку.

9.5. Функция mainWebSocketSend

Функция mainWebSocketSend, реализованная в файле «dynrobot.js», отправляет в программный комплекс «ДинРобот-3» указанное текстовое сообщение. При этом если главный WebSocket находится в подключенном состоянии, то отправка производится немедленно, а если в неподключенном – то сообщение ставится в очередь на отправку. Таким образом, разработчик контента может не беспокоится о доставке своего сообщения.

Однако, очередь на отправку актуальна в течение 1.5 секунд. Далее сообщения из нее удаляются. Считается, что если за это время соединение не восстановилось, значит связь окончательно разорвана, а само сообщение через 1.5 секунды станет уже не актуальным.

```
mainWebSocketSend(msg);
```

Где:

msg – текстовое сообщение, отправляемое в «ДинРобот-3».

Например:

```
mainWebSocketSend("say Привет мир");
```

9.6. Функция isMainWebSocketConnected

Функция isMainWebSocketConnected, реализованная в файле «dynrobot.js», проверяет, находится ли главный WebSocket в подключенном состоянии. Это позволяет скрипту Web-контента не отправлять сообщения с критической актуальностью, если в данный момент нет связи с «ДинРобот-3». Напомним, что сообщения, отправляемые функцией mainWebSocketSend, ставятся в очередь на отправку, если главный WebSocket не подключен. Но некоторые команды, например, команды управления шасси, нельзя ставить в очередь на отправку. Их лучше не исполнить, чем исполнить с задержкой.

```
isMainWebSocketConnected();
```

Например:

```
if (isMainWebSocketConnected())
{
   mainWebSocketSend("iscript:put * myEvent");
}
```

9.7. Получение состояния синтезатора речи (говорит ли робот)

В «dynrobot.js» есть глобальная переменная isSpeaking, которая принимает значение true, либо false в зависимости от того, говорит ли в данный момент робот или нет. Обработчик сообщений от главного WebSocket самостоятельно отслеживает сообщения «speech:state» от «ДинРобот-3» и выставляет значение этой переменной.

Разработчику скриптов следует понимать, что переменная isSpeaking не мгновенно становится равной true после вызова функции playSpeech или playSpeechLang. Должно пройти небольшое время с момента передачи сообщения в «ДинРобот-3» и моментом получения от него сообщения о состоянии синтезатора речи. Поэтому если скрипт пользователя собрался «пулить» состояние этой переменной по таймеру, необходимо сначала дождаться, когда переменная станет true, и лишь за тем, ждать ее перехода в false.

Например, говорить каждые 20 секунд фразу «Привет мир», при условии, что робот не говорит в этот момент еще что-нибудь:

```
<html>
<body>
<script src="contentui.js"></script>
<script>
setInterval(function()
  {
      if (!isSpeaking) playSpeech("Привет мир");
  }, 20000);
</script>
</body>
</html>
```

Рекомендуется использовать переменную isSpeaking лишь для тех случаев, когда скрипту Web-контента нужно лишь определить, не говорит ли робот что-то в тот момент, а не для отслеживания окончания собственной речи.

Для отслеживания состояния собственной речи в файле «dynrobot.js» реализован механизм вызова callback-функции «onSpeechOver», объявленной глобальной переменной и имеющей состояние null. Функция вызывается всякий раз, когда синтезатор речи замолкает (приходит сообщение «speech:state 0»).

Разработчик скрипта может присвоить переменной onSpeechOver указатель на какую-то свою функцию, чтобы отловить с помощью нее событие окончания речи.

Например:

```
<html>
<body>
<div id=stateDiv>Робот молчит</div>
<button onclick="doSpeaking()">Нажмите, чтобы сказать</button>

<script src="contentui.js"></script>
<script>

// функция вызывается при нажатии кнопки
function doSpeaking()
{
   playSpeech("Привет мир"); // начать говорить
   stateDiv.innerHTML = "Робот говорит"; // изменить состояние

   // подписаться на вызов функции onSpeechOver - функция будет
   // вызвана по окончанию синтеза речи
   onSpeechOver = function()
   {
     onSpeechOver=null; // отписаться от onSpeechOver
     stateDiv.innerHTML = "Робот молчит"; // изменить состояние
   }
}
</script>
</body>
</html>
```

Сам же скрипт в «dynrobot.js» отслеживает состояние синтезатора речи по сообщениям от «ДинРобот-3»: «speech:state 1» (робот говорит) и

«speech:state 0» (робот замолчал), отправленных в Web-контент через главный WebSocket.

9.8. Получение сообщений от программного комплекса «ДинРобот-3»

В файле «dynrobot.js» объявляется глобальная переменная **mainWebSocketListeners**, представляющая собой массив подписчиков на получение сообщений от главного WebSocket, т.е. массив указателей на функции получения сообщений от программного комплекса «ДинРобот-3».

Для получения сообщений от программного комплекса «ДинРобот-3» необходимо, так сказать, подписаться на рассылку. Для этого нужно объявить функцию-обработчик сообщений и добавить ссылку на нее в массив mainWebSocketListeners. На одной Web-странице может быть несколько таких подписчиков, объявленных, например, в разных независимых друг от друга скриптах.

Одна функция-обработчик может обрабатывать несколько разных сообщений, если ей это необходимо. Главное, функция-обработчик должна самостоятельно разобрать текст сообщения и идентифицировать в нем обрабатываемое сообщение. Удобно это делать с помощью стандартного метода match по регулярному выражению.

Сама функция-обработчик должна вернуть true, если она обработала сообщение, тем самым запретив другим обработчикам обработать это сообщение, включая главный обработчик в «dynrobot.js».

Однако, обработав сообщение, функция-обработчик не обязательно должна возвращает true. Она может вернуть и false, если обработанное сообщение может понадобиться еще какому-нибудь подписчику или главному обработчику.

На вход функции поступает текст сообщения, отправляемого из «ДинРобот-3» через главный WebSocket.

Например:

```
<html>
<body>
<script src="contentui.js"></script>

<div id=resultDiv>-</div>
<script>
// функция-обработчик сообщений от «ДинРобот-3» (подписчик)
function myHandler(msg)
{
   // разобрать и идентифицировать сообщение по регулярному выражению
   var g = msg.match(/^mimic (.)/);
   if (g)
   {
      resultDiv.innerHTML = g[1]; // вывести символ мимики из сообщения
      return true; // сообщить, что сообщение обработано
   }
   return false; // сообщить, что сообщение не обработано
}
mainWebSocketListeners.push(myHandler); // добавить ссылку на
```

```
                                // функцию-обработчик в массив подписчиков
                                // на рассылку сообщений от «ДинРобот-3»
</script>
</body>
</html>
```

9.9. Включение и выключение режима рассылки сообщений с мимикой лица

Под мимикой лица подразумевается поток фонем и других эмоций для реализации анимации лица робота. Мимика актуальна, если лицо робота по тем или иным причинам реализуется на главном экране робота.

Сообщения мимики отправлятся через главный WebSocket от программного комплекса «ДинРобот-3» в экранный Web-контент. Но такие сообщения не отправляются по умолчанию, режим рассылки таких сообщений необходимо включать и подтверждать через каждые 1-2 секунды. Если через 2.5 секунды Web-контент не подтвердит необходимость получения таких сообщений, передача будет отключена.

Для включения и подтверждения режима рассылки мимики необходимо через главный WebSocket отправить текстовое сообщение **«mimic:on»**.

Для отключения режима рассылки мимики необходимо через главный WebSocket отправить текстовое сообщение **«mimic:off»** или 2.5 секунды не подтверждать режим рассылки (лучше, конечно, режим рассылки явно выключить).

Например:

```
<html>
<body>
<script src="contentui.js"></script>
<div id=resultDiv>-</div>
<script>

// функция-обработчик сообщений от «ДинРобот-3» (подписчик)
function myHandler(msg)
{
   // разобрать и идентифицировать сообщение по регулярному выражению
   var g = msg.match(/^mimic (.)/);
   if (g)
   {
      resultDiv.innerHTML = g[1]; // вывести символ мимики из сообщения
      return true; // сообщить, что сообщение обработано
   }
   return false; // сообщить, что сообщение не обработано
}
mainWebSocketListeners.push(myHandler); // добавить ссылку на
                                // функцию-обработчик в массив подписчиков
                                // на рассылку сообщений от «ДинРобот-3»
// функция включение и подтверждение режима мимики
function setMimicOn()
{
   mainWebSocketSend("mimic:on");
}
setInterval(setMimicOn, 1000); // каждую секунду вызывать setMimicOn
setMimicOn(); // включить режим рассылки мимики
```

```
</script>
</body>
</html>
```

9.10. Сообщения от «ДинРобот-3» в экранный контент

9.10.1. Сообщение о состоянии синтезатора речи (speech:state)

Сообщение «speech:state 1» отправляется от программного комплекса «ДинРобот-3» в экранный Web-контент, если робот начал что-то говорить.

Сообщение «speech:state 0» отправляется от программного комплекса «ДинРобот-3» в экранный Web-контент, если робот закончил говорить.

9.10.2. Сообщение с кодом на javascript (url javascript://…)

Сообщение «url javascript://*js-код*» отправляется от программного комплекса «ДинРобот-3» в экранный Web-контент при вызове функции в IScript3: «SetBrowserURL("javascript://*js-код*")», где: *js-код* – программный код на языке JavaScript.

Таким образом, из IScript3 можно управлять состоянием какой-либо Web-страницы. Например, голосовой командой, распознанной через IScript3, нажать кнопку в экранном контенте.

Следует учитывать, что IScript3 использует кодировку ANSI (в России это Windows-1251), а экранный контент использует кодировку UTF-8. Поэтому, если функция SetBrowserURL пытается передать в JavaScript строку текста, в которой присутствуют русские буквы, то необходимо сначала эту строку перевести в кодировку UTF-8 (с помощью IScript3-функции ANSIToUTF8).

Например, по голосовой команде «Красный», «Зеленый» или «Синий» изменяем цвет фона экрана.

Код HTML-станицы:

```
<html><body><script src="contentui.js"></script></body></html>
```

Код в IScript3:

```
// Создать командный фреймсет colors
frameset("colors",1)
{
   frame("Красный") // по голосовой команде «красный»...
   {
      SetBrowserURL("javascript://document.body.bgColor=#ff0000");
   }
   frame("Зеленый") // по голосовой команде «зеленый»...
   {
      SetBrowserURL("javascript://document.body.bgColor=#00ff00");
   }
   frame("Синий")  // по голосовой команде «синий»...
   {
      SetBrowserURL("javascript://document.body.bgColor=#0000ff");
   }
}
SpeechRecognizer(true);  // включить распознавание речи
while(true) sync();      // вечный главный цикл скрипта
```

Сам же «dynrobot.js» при получении сообщения «url javascript://…» выполняет код с помощью метода window.eval(…).

9.10.3. Сообщение о переходе на указанный URL (url …)

Сообщение «url *адрес*» отправляется от программного комплекса «ДинРобот-3» в экранный Web-контент при вызове функции в IScript3: «SetBrowserURL("*адрес*")», где: *адрес* – URL-адрес страницы, куда необходимо перейти экранному контенту.

Используя данный механизм, IScript3 может переключать страницы экранного контента. Например, по голосовой команде можно открыть ту или иную страницу экранного контента.

Сам же «dynrobot.js» при получении сообщения «url …» переходит на страницу путем вызова «location=…».

9.10.4. Сообщения о мимике лица робота (mimic …)

Сообщение «mimic *код период*» отправляется от программного комплекса «ДинРобот-3» в экранный Web-контент при установки очередной фонемы или изменении эмоции лица робота.

Здесь:

«код» – код мимики (латинские буквы):

«A» – код мимики буквы «а» («я).
«O» – код мимики буквы «о» («ё»).
«U» – код мимики буквы «у» («ю»).
«E» – код мимики буквы «е» («э»).
«0» – код нейтральной мимики рта, согласная буква.
«)» – улыбка.
«(» – грусть.
«|» – нейтральное выражение лица.
«^» – поднять брови.
«<» – посмотреть немного глазами влево.
«>» – посмотреть немного глазами право.
«B» – моргание.

«период» – период (мс), за который необходимо перейти в данное состояние.

Как уже говорилось ранее (см. п. 9.9), для рассылки сообщений с фонемами лица необходимо через главный WebSocket включить режим «mimic:on» и постоянного его подтверждать.