Четыре типа REST-методов

Существует четыре типа REST-методов:

1. connect
2. прямые вызовы
3. события
4. входящие вызовы

Метод `connect` определяет поведение всех остальных методов.

Метод `connect` должен возвращать в ответе ровно те данные, которые он получил от WCS, за исключением тех случаев, когда нужно явно переопределить какое-либо поле ответа либо нужно явно вернуть другой статус.

Остальным методам рекомендуется возвращать в ответе ровно те данные, которые они получили, за исключением тех случаев, когда нужно явно переопределить какое-либо поле ответа либо нужно явно вернуть другой статус.

Все методы должны возвращать HTTP-статус `200 OK`, за исключением тех случаев, когда требуется явно вернуть другой статус, например статус ошибки `403 Forbidden`.

Прямые вызовы со стороны WCS JavaScript API

Эти вызовы могут быть прерваны бэкенд сервером:

- `call` - исходящий звонок может быть запрещен
- `answer` - можно запретить ответить на звонок
- `hold` - звонок можно запретить ставить на удержание
- `unhold` - звонок можно запретить снимать с удержания
- `transfer` - можно запретить делать трансфер звонка
- `sendDTMF` - можно запретить отправку DTMF
- `sendMessage` - можно запретить отправку исходящего сообщения
- `sendIMDN` - можно запретить отправку подтверждения доставки
- `subscribe` - можно запретить SIP-подписку на события
- `sendXcapRequest` - можно запретить отправку XCAP
- `publishStream` - можно запретить публикацию потока

- `playStream` - можно запретить воспроизведение потока
- `playHLS` - можно запретить воспроизведение потока по HLS
- `playRTSP` - можно запретить воспроизведение потока по RTSP
- `OnDataEvent` - web-сервер может не принять данные
- `sendBugReport` - можно запретить отправку багрепорта

Прерывание этих вызовов не запрещено, но нецелесообразно:

- `hangup` - звонок можно запретить сбрасывать
- `unPublishStream` - можно запретить остановку публикации потока
- `stopStream` - можно запретить остановку воспроизведения потока

События, которые уже произошли и не могут быть запрещены бэкенд сервером

Эти вызовы не могут быть прерваны бэкенд сервером, т.к. соответствующее событие уже произошло на стороне WCS, и бэкенд серверу направляется только уведомление.

- `ConnectionStatusEvent`
- `RegistrationStatusEvent`
- `CallStatusEvent`
- `TransferStatusEvent`
- `MessageStatusEvent`
- `SubscriptionStatusEvent`
- `XcapStatusEvent`
- `StreamStatusEvent`
- `DataStatusEvent`
- `BugReportStatusEvent`
- `StreamKeepAliveEvent`
- `sendStreamEvent`
- `StreamEvent`

Входящие вызовы

Входящие вызовы могут быть прерваны бэкенд сервером. В случае прерывания входящего вызова WCS уведомляет об этом инициатора вызова.

- `OnCallEvent` - входящий звонок

- `OnMessageEvent` - входящее сообщение
- `OnTransferEvent` - входящий трансфер

Обработка вызовов, инициированных при помощи REST API

Если вызов REST-метода любого типа был инициирован при помощи REST API, он не должен быть отклонен бэкенд сервером

Действия в случае ошибки взаимодействия с бэкенд сервером

Ошибкой взаимодействия считается возврат бэкенд сервером статуса, отличного от `200 OK` или ошибка, которая препятствует обращению к бэкенд серверу. В зависимости от поля `restOnError` в restClientConfig и типа метода будут произведены следующие действия, указанные в таблице:

restOnError	**Connect**	**Direct invocations**	**Events**	**Incoming calls**
FAIL	1. Логгируем ошибку 2. Прерываем выполнение 3. Спускаем ошибку на клиента в соответствующем вызову событии	1. Логгируем ошибку 2. Прерываем выполнение 3. Спускаем ошибку на клиента в соответствующем вызову событии	1. Логгируем ошибку 2. Продолжаем выполнение 3. Спускаем ошибку на клиента в специальном событии `ErrorEvent`	1. Логгируем ошибку 2. Прерываем выполнение 3. Отвечаем инициатору вызова статусом `403 FORBIDDEN` 4. Спускаем ошибку на клиента в соответствующем вызову событии
LOG		1. Логгируем ошибку 2. Продолжаем выполнение	1. Логгируем ошибку 2. Продолжаем выполнение	1. Логгируем ошибку 2. Продолжаем выполнение

Действия в случае других ошибок

В случае получения SIP статусов 4xx, 5xx, 6xx, а так же в случае других ошибок, не связанных с REST, будут инициированы соответствующие события со статусом

`FAILED` и описанием в поле `info`. Эти события будут отправлены бэкенд серверу, а затем клиенту в соответствии с правилами restClientConfig.

Например, если при исходящем звонке SIP сервер вернет `403 FORBIDDEN`, на бэкенд сервер и на клиента будет отправлено `CallStatusEvent` `status: "FAILED", info:"SIP 403 FORBIDDEN", sipMessageRaw:"исходное SIP сообщение"`.

Если ошибку нельзя соотнести ни с одним из существующих событий, то будет инициировано событие ErrorEvent с описанием ошибки в поле `info`.

Рассмотрим по одному методу от каждого типа (остальные методы ведут себя одинаково внутри типа).

Тип 1 - connect

Тип 2 - прямой вызов

Тип 3 - событие

Тип 4 - входящий вызов