# ПРИЛОЖЕНИЕ 6

к Альбому ТФФ версии 6.0

|  |  |  |
| --- | --- | --- |
| ФЕДЕРАЛЬНОЕ КАЗНАЧЕЙСТВО | | |
|  |  |  |
|  |  |  |

**ОПИСАНИЕ ПРОТОКОЛА УСТОЙЧИВОГО К ОШИБКАМ СОЕДИНЕНИЯ ПРИЕМА ФАЙЛОВ**

#### Общее описание компонента.

Данный компонент реализует протокол устойчивого к ошибкам соединения приема файлов (resumable upload). Протокол реализуется поверх протокола HTTP 1.1, то есть использует методы и заголовки (как стандартные, так и, возможно, свои специфичные) протокола HTTP для реализации возобновляемой, устойчивой к ошибкам передачи файлов. Использование протокола HTTP over TLS 1.0 / SSL 3.0 в промышленной эксплуатации обязательно (так как используется аутентификация типа Basic и идентификационная информация клиента передается в открытом виде).

#### Описание протокола

Протокол состоит из следующих основных стадий (взаимодействий)

1. **Начало сессии возобновляемой передачи файла**. Данным взаимодействием (запросом) клиент извещает сервер о своем намерении начать передачу файла. Клиент сообщает некоторые метаданные о файле, но НЕ посылает содержимое файла. При успешном создании сессии сервер в HTTP ответе возвращает клиенту URI, который необходимо использовать при передаче данных файла и идентификатор файла в Хранилище Файлов.
2. **Передача данных файла/части файла (повторяется необходимое число раз)**. В данном взаимодействии клиент, используя полученный сессионный URI, передает содержимое файла/части файла на сервер. В ответ сервер сообщает статус загрузки файла.
3. **Запрос текущего статуса передачи файла**. С помощью данного взаимодействия клиент в любой момент может узнать статус передачи файла. Особенно это полезно в случае проблем с сетевым соединением, проблем с доступностью сервиса и проблем с работоспособностью самого клиента. В результате сервер сообщает статус передачи:
   1. Передача уже завершена, то есть файл полностью передан.
   2. Передача не завершена. В этом случае сервер сообщает, какая часть содержимого файла уже принята (диапазон байтов)
   3. Неизвестная сессия передачи. Сессия передачи файла с указанным идентификатором неизвестна серверу.
4. **Завершение передачи файла**. Клиент извещает сервер о том, что он закончил (считает законченной) передачу файла.

**Ограничения**:

Установлены ограничения на минимальный и максимальный размер куска: 10 Кб минимум, 1 Мб максимум. Контроль этих ограничений должен происходить как можно "раньше", идеально - на фронтальных серверах. Верхняя граница размера куска также является верхней границей размера данных, передаваемых внутри пакета альтернативной интеграции (на переходный период).

#### Аутентификация клиента

При каждом обращении к сервису передачи файлов клиент должен быть аутентифицирован. Клиент должен быть готов к тому, что на любой запрос клиента сервер может вернуть ответ следующего вида

**Ошибка аутентификации**

|  |
| --- |
| HTTP/1.1 401 Authorization Required  WWW-Authenticate: Basic realm="file-storage" |

Это означает, что сервер не смог аутентифицировать клиента и клиент обязан выполнить аутентификацию.

Протокол поддерживает аутентификацию клиента с помощью стандартной HTTP Basic Authentication (смотри RFC 2617). Basic Authentication предполагает добавление заголовка Authorization в HTTP запрос. Заголовок имеет следующий формат:

Authentication\_header = "Authorization: Basic " + user\_credentials

user\_credetials = Base64Encode(login + ":" + password)

где:

* login - идентификатор клиента
* password - пароль клиента
* '+' - операция конкатенации строк
* Base64Encode(...) - функция кодирования байтов в Base64

При получении от клиента запроса, содержащего идентификационную информацию, сервер проводит аутентификацию. В случае успешной аутентификации сервер выполняет действия, предусмотренные запросом клиента и возвращает токен аутентификационной сессии, который может быть сохранен клиентом для последующих обращений к серверу без указания аутентификационной информации и, следовательно, без выполнения сервером аутентификационных действий на каждый запрос. Токен передается клиенту в составе стандартного HTTP заголовка cookies в следующем формате:

LtpaToken2=<значение-токена>

где <значение-токена> - длинная строка из различных символов, в которой закодирована аутентификационная информация.

Клиент может использовать 2 основных сценария аутентификации

**Аутентификация без создания аутентификационной сессии.**

Клиент не поддерживает по каким-то причинам cookie и поэтому передает аутентификационную информацию в каждом запросе. Сценарий простой, не требует от клиента поддержки cookie, но требует дополнительных ресурсов сервера на выполнение аутентификации при каждом запросе клиента.

**Аутентификация с созданием аутентификационной сессии.**

При аутентификации создается аутентификационная сессия, то есть сервер генерирует токен специального вида, по которому сервер может "опознать" данного клиента, не проводя аутентификацию. Сессия имеет некоторое время жизни. Следовательно, клиент должен быть готов к тому, что время жизни аутентификационной сессии может закончится в любой момент и провести повторную аутентификацию.

Алгоритм работы клиента в сценарии с аутентификационной сессией.

1. Неаутентифицированный клиент в любом запросе к серверу указывает идентификационную информацию.
2. После успешной аутентификации сервер в ответном сообщении указывает токен аутентификационной сессии.
3. Клиент запоминает токен аутентификационной сессии
4. Клиент передает токен аутентификационной сессии серверу в каждом последующем запросе.

Клиент в любой момент может завершить текущую аутентификационную сессию / создать новую аутентификационную сессию, передав серверу запрос, который:

1. не содержит токена текущей аутентификационной сессии
2. содержит аутентификационную информацию

### Формат взаимодействий (запросов-ответов) протокола

#### Начало сессии возобновляемой передачи файла

Для начала сессии возобновляемой передачи файла необходимо отправить запрос на URI создания сессии передачи (определяется на этапе развертывания компонентов) следующего формата:

* HTTP метод - POST
* Тело запроса, которое содержит метаинформацию о файле в формате JSON, в кодировке UTF-8, в составе следующих атрибутов
  1. name - имя файла
  2. size - размер файла в байтах
  3. digest - хэш-сумма содержимого файла, рассчитанная по алгоритму SHA-256. Используется для контроля корректности переданных данных.
* Заголовок Content-Type содержит следующее значение: "application/json; charset=UTF-8"
* Заголовок Content-Length содержит размер тела запроса.

Пример запроса приведен ниже

**Запрос инициации сессии передачи файла**

|  |
| --- |
| POST {create-session-uri} HTTP/1.1  Authorization: Basic dGVzdDoxMjNxd2VBU0Q=  Content-Length: 143  Content-Type: application/json; charset=UTF-8  {"name":"Мой тестовый файл.docx","size":15098192,"digest":"bMcmsuV07MJB49gMKELBFCnso4+Y9dagSgZUUF4mLf0\u003d"} |

В случае успешного создания сессии передачи файла сервер возвращает ответ следующего формата:

* HTTP статус - 200 (ОК)
* Заголовок Location содержит URI, который должен использоваться для передачи данных файла.
* Тело запроса, которое содержит Идентификатор Содержимого Файла в формате JSON, в кодировке UTF-8.
* Заголовок Content-Type содержит следующее значение: "application/json; charset=UTF-8"
* Заголовок Content-Length содержит размер тела запроса.

Пример "успешного" ответа сервера приведен ниже

**Успешный ответ сервера**

|  |
| --- |
| HTTP/1.1 200 OK  Location: {upload-file-session-uri}  Content-Length: 58  Content-Type: application/json; charset=UTF-8  {"file\_content\_id":"be9d6f42-2d6d-4ab7-ab68-89b24fd2b922"} |

В случае, если файл с указанным содержимым уже присутствует в Хранилище файлов, сервер возвращает ответ следующего формата:

* HTTP статус - 201 (Created)
* Тело запроса, которое содержит Идентификатор Содержимого Файла в формате JSON, в кодировке UTF-8.
* Заголовок Content-Type содержит следующее значение: "application/json; charset=UTF-8"
* Заголовок Content-Length содержит размер тела запроса.

**Успешный ответ сервера в случае если файл уже есть**

|  |
| --- |
| HTTP/1.1 201 Created  Content-Length: 58  Content-Type: application/json; charset=UTF-8  {"file\_content\_id":"be9d6f42-2d6d-4ab7-ab68-89b24fd2b922"} |

При этом сессия НЕ создается.

#### Передача данных файла/части файла

Клиент посылает запрос следующего формата:

* HTTP метод - POST
* Запрос идет на URI сессии передачи файла
* Заголовок Content-Length содержит длину части файла в байтах
* Заголовок Content-Type содержит "application/octet-stream"
* Заголовок Content-Range содержит указание диапазона байтов (индексы первого и последнего байта) содержимого файла, который передается в данной части. Так же указывается полный размер файла в байтах.
* Тело запроса содержит байты из указанного в заголовке Content-Range диапазона.

Пример запроса приведен ниже

**Запрос на передачу части файла**

|  |
| --- |
| POST {upload-file-session-uri} HTTP/1.1  Host: zakupki.gov.ru  Authorization: Basic dGVzdDoxMjNxd2VBU0Q=  Content-Length: 512000  Content-Type: application/octet-stream  Content-Range: bytes 0 - 512000/15098192  {bytes from 0 to 512000} |

Сервер, в зависимости от того что произошло, может ответить следующим образом

При успешной передача части файла сервер присылает ответ следующего формата

* HTTP код - 202 Accepted
* Заголовок Range содержит диапазон байтов передаваемого файла (не части, а всего файла), который сервер уже принял на данный момент.

Пример такого ответа приведен ниже

**Ответ в случае успешной передачи части файла**

|  |
| --- |
| HTTP/1.1 202 Accepted  Content-Length: 0  Range: 0-512000 |

В случае, если серверу неизвестна сессия (запрос сделан на URI, по которому не может производится передача файла)

**Ошибка - неизвестная сессия**

|  |
| --- |
| HTTP/1.1 404 Not Found  Content-Length: 0 |

#### Запрос текущего статуса передачи файла

Реализуется запросом следующего формата:

* Метод HTTP - POST
* Запрос идет на URI сессии передачи файла.
* Заголовок Content-Range содержит неопределенный диапазон. Неопределенный диапазон записывается в виде
  + "\*/{полный-размер-файла-в-байтах}", если известен полный размер файла.
  + **"\*/\*"**, если полный размер файла не известен.

Пример запроса приведен ниже (полный размер файла известен)

**Запрос текущего статуса передачи файла**

|  |
| --- |
| POST {upload-file-session-uri} HTTP/1.1  Authorization: Basic dGVzdDoxMjNxd2VBU0Q=  Content-Length: 0  Content-Range: bytes \*/1000000 |

Сервер может вернуть ответ одного из двух типов

**Ответ о незавершенной передаче файла**

Пример такого ответа приведен ниже:

**Ответ о незавершенной передаче файла**

|  |
| --- |
| HTTP/1.1 202 Accepted  Content-Length: 0  Range: 0-512000 |

**Ошибка "Неизвестная сессия передачи файла"**

В случае, если серверу неизвестна сессия (запрос сделан на URI, по которому не может производится передача файла)

**Ошибка - неизвестная сессия**

|  |
| --- |
| HTTP/1.1 404 Not Found  Content-Length: 0 |

#### ****Завершение передачи файла****

Реализуется запросом следующего формата:

* HTTP метод - POST
* Запрос идет на URI сессии передачи файла
* Заголовок Content-Length содержит длину тела запроса, то есть строки параметров
* Заголовок Content-Type содержит значение "application/x-www-form-urlencoded; charset=UTF-8"
* Тело запроса содержит параметр запроса в форме status=completed, как указано в примере

Пример запроса приведен ниже

**Запрос, извещающий о окончании загрузки файла**

|  |
| --- |
| POST {upload-file-session-uri} HTTP/1.1  Host: zakupki.gov.ru  Content-Length: <content-length>  Content-Type: application/x-www-form-urlencoded; charset=UTF-8  status=completed |

В ответ сервер может вернуть один из следующих ответов

**Файл корректно передан**

Если файл получен сервером полностью и контрольная сумма совпадает с той, что была передана при создании сессии

**Ответ в случае если загрузка файла полностью завершена**

|  |
| --- |
| HTTP/1.1 HTTP 201 Created  Content-Length: 0 |

**Файл передан не полностью**

Клиент указал на то, что он завершил передачу файла, но сервер не получил еще всех данных. При этом сервер указывает, какие данные он получил на данный момент

**Ответ о незавершенной передаче файла**

|  |
| --- |
| HTTP/1.1 202 Accepted  Content-Length: 0  Range: <диапазон байтов, полученных сервером> |

**Ошибка "Неверная Контрольная сумма"**

**Неверная контрольная сумма файла**

|  |
| --- |
| HTTP/1.1 409 Conflict  Content-Length: 58  Content-Type: application/json; charset=UTF-8  {"digest\_expected":"bMcmsuV07MJB49gMKELBFCnso4+Y9dagSgZUUF4mLf0\u003d", "digest\_actual":"1McmsuV07MJB49gMKELBFCnso4+Y9dagSgZUUF4mLf0\u003d"} |

**Ошибка "Неизвестная сессия передачи файла"**

В случае, если серверу неизвестна сессия (запрос сделан на URI, по которому не может производится передача файла)

**Ошибка - неизвестная сессия**

|  |
| --- |
| HTTP/1.1 404 Not Found  Content-Length: 0 |