SMVision

версия протокола 1.4
редакция 018

Оглавление

1. Общие принципы

1.1 Назначение протокола

Протокол предназначен для реализации сервиса видео распознавания для конечных устройств (клиентов). В качестве клиентов могут выступать весы самообслуживания, кассовые терминалы, киоски самообслуживания, оборудованные видео камерой, весовой платформой и сетевым интерфейсом.

1.2 Логика протокола

Логика протокола реализуется путём выполнения набора операций в различных сценариях. Каждая операция выполняется путём взаимодействия клиентов с сервером видео распознавания с использованием RestAPI. В рамках протокола реализуются следующие операции:
1) recognize - распознавание товара
2) upload – сохранение данных о выборе покупателя

В рамках сценариев могут выполняться несколько операций объединённых логическим контекстом (сеансом). Принадлежность операций к сеансу определяется путём использования общего идентификатора сеанса (**session_id**). Каждая операция должна иметь уникальный идентификатор операции (**request_id)** .

Уникальность **session_id** и **request_id** должна поддерживаться в рамках всей системы SMVision (среди всех операций всех клиентов) путём применения алгоритма **UUID**.
Примечание: Алгоритм UUID подразумевает коллизии, но вероятность коллизии двух случайно сгенерированных UUID4 крайне мала (10^-38 в случае идеального генератора случайных чисел).

1.3 Форматы данных

1.3.1 Кодировка

Кодировка строковых полей для всех операций: `UTF-8`

1.3.2 Общий формат запросов/ответов

Формат запроса

http header (заголовок запроса): набор параметров (Name: Value) в формате простых строк
http body (тело запроса): multipart/form-data – набор бинарных данных (фото товара) и параметров в виде Json строк или простых строк вида (Name=Value)
где

Name – название параметра
Value – значение параметра

Формат ответа

http status (статус ответа): простые строки
http header (заголовок ответа): простые строки
http body (тело ответа): Json строки

Простая строка – строка, не имеющая внутреннего форматирования или структуры

Json строка – строка, содержащая структурированные данные в формате Json

1.3.3 Типы данных

UNQUOTED_STRING - строка без кавычек

Пример:

```
12345678-09A2-4348-AE91-7E1FA26E5E3A
```

STRING - строка, заключённая в кавычки

Пример:

```
"12345678-09A2-4348-AE91-7E1FA26E5E3A"
```

BOOLEAN_STRING - булева строка (применяется в простых строках)

Значения:

```
"true"
"false"
```

Пример (из запроса “upload” / body):

```
"dry_run": "false"
```

JSON_BOOLEAN - булев тип Json, применяется в Json строках

Значения:

```
true
false
```

Пример (из запроса “upload” / body)

```
"non_weight": false
```

JSON_INTEGER - целое число, применяется в Json строках

JSON_FLOAT - число с плавающей точкой, применяется в Json строках

JSON_NUMBER - число (JSON_INTEGER или JSON_FLOAT), применяется в Json строках

JSON_STRUCT - Json структура

Пример

```
{
      "name": "Картофель фасованный",
      "sku": "1036123",
      "score": 0.22121392800343484
},
```

JSON_ARRAY - Json массив

Пример

```
"result":
 [
      {
            "name": "Картофель фасованный",
            "sku": "1036123",
            "score": 0.22121392800343484
      },
      {
            "name": "Гранат",
            "sku": "1036110",
            "score": 0.14674207251916002
      },
      {
            "name": "Яблоко Красное",
            "sku": "1036175",
            "score": 0.02716088717745504
      },
      {
            "name": "Лук репчатый",
            "sku": "1036133",
            "score": 0.017922750001051348
      },
      {
            "name": "Грейпфрут",
            "sku": "1036111",
            "score": 0.01113253679277089
      }
]
```

1.4 Сценарии

1.4.1 Виды сценариев

В рамках протокола реализуются следующие сценарии:
1) Сбор данных
2) Форсированный сбор данных
3) Распознавание
4) Без распознавания

Клиенты должны обеспечивать режимы работы, соответствующие каждому из указанных сценариев.

1.4.2 Сбор данных

Сценарий «Сбор данных» предназначен для неявного сбора данных о товарах без распознавания .
Сценарий предназначен для накопления данных о товарах в фоновом режиме. Сценарий применяется на этапе внедрения системы видео распознавания. Сценарий активируется автоматически в фоновом режиме по мере того, как покупатели взвешивают товары и выбирают их с использованием пользовательского интерфейса клиента. При этом фотографирование товара и передача информации о товаре на сервер выполняется клиентом автоматически без участия покупателя.
Сценарий состоит из следующих фаз:
а) операция “upload” - обязательно однократно
б) операция “upload” - опционально (количество неограниченно)

Фаза **б** возможна в ситуации, когда покупатель после первоначального выбора товара на пользовательском интерфейсе клиента, изменил выбор товара, не убирая товар с весовой платформы.

Пример
Покупатель поставил товар на весы
Выбрал **товар_1** (фаза **а**)
Не убирая товар с платформы, выбрал другой товар **товар_2** (фаза **б**).

На сервере видео распознавания действительной считается последняя операция upload.
Для параметра “`forced_dataset_collection`” должно быть установлено значение “`false`”.

1.4.3 Форсированный сбор данных

Сценарий «Форсированный сбор данных» предназначен для явного сбора данных о товарах без распознавания. Сценарий предназначен для накопления данных о товарах в режиме обучения системы видео распознавания, когда пользователь намеренно выбирает определённые товары для явной привязки изображения товара к информации о товаре в системе видео распознавания. Сценарий может применяться как на этапе внедрения, так и в ходе эксплуатации системы видео распознавания.

Сценарий состоит из следующих фаз:
а) операция “upload” - обязательно однократно
б) операция “upload” - опционально (аналогично фазе “**б**” из сценария «Сбор данных»)

Для параметра “`forced_dataset_collection`” должно быть установлено значение “`true`”.
Данный сценарий аналогичен сценарию «Сбор данных» за исключением значения параметра “`forced_dataset_collection`“.

1.4.4 Распознавание

Сценарий «Распознавание» предназначен для распознавания товаров в системе, для которой ранее применялись сценарии «Сбор данных» или «Форсированный сбор данных».

Сценарий «Распознавание» состоит из следующих фаз:
а) операция “recognize” - обязательно однократно
б) операция “recognize” - опционально (количество неограниченно)
в) операция “upload” - обязательно однократно
г) операция “upload” - опционально (аналогично фазе “**б**” из сценария «Сбор данных»)

1.4.5 Без распознавания

Данный сценарий применяется в случае, когда покупатель выбирает товар до того, как поставит товар на весовую платформу. Сценарий «Без распознавания» схож со сценарием «Сбор данных» и отличается тем, что данный сценарий возможен после этапа начального внедрения, в отличии от сценария «Сбор данных».

Сценарий «Без распознавания» состоит из следующих фаз:

а) операция “upload” - обязательно однократно

б) операция “upload” - опционально (количество неограниченно)

2. Операции

2.1 Операция “recognize”

2.1.1 Описание операции

Операция “recognize” реализуется путём отправки на сервер POST запроса “recognize” с последующим получением ответа, содержащего результат распознавания. Данный запрос хранит в себе:

1) изображение товара для распознавания;
2) дополнительные параметры, используемые сервером.

2.1.2 Запрос

2.1.2.1 Структура запроса

Возможные шаблоны URL запроса:

1) `https://`**`HOSTNAME`**`/recognize/`**`PARTNER_ID`**
2) `http://IP_ADDRESS:`**`PORT`**`/recognize/`**`PARTNER_ID`**

`IP_ADDRESS` – IP адрес сервера видео распознавания
`PORT` – TCP port сервиса видео распознавания
`HOSTNAME` – хост сервера видео распознавания
`PARTNER_ID` – идентификатор партнёра (торговой сети, использующей сервис видео распознавания)

Запрос recognize представляет собой http multipart/form-data контейнер, состоящий из 3 частей:
1) header
2) body(text)
3) body(binary)

2.1.2.2 header

header состоит из следующего набора текстовых полей

```
vision-protocol-version: VISION_PROTOCOL_VERSION_UNQUOTED_STRING
vision-token: VISION_TOKEN_UUID_UNQUOTED_STRING
```

Описание полей

VISION_PROTOCOL_VERSION_UNQUOTED_STRING - версия протокола
значение: 1.4

VISION_TOKEN_UUID_UNQUOTED_TRING – идентификатор конечного устройства
значение: UUID без скобок

2.1.2.3 body(text)

body(text) состоит из следующего набора текстовых полей

```
"session_id": SESSION_ID_UUID_STRING
"request_id": REQUEST_ID_UUID_STRING
"product_image_md5": PRODUCT_IMAGE_MD5_STRING
"dry_run": DRY_RUN_BOOLEAN_STRING
```

Описание полей

SESSION_ID_UUID_STRING - идентификатор сеанса(строка)
значение: UUID без скобок
REQUEST_ID_UUID_STRING - идентификатор запроса(строка)
значение: UUID без скобок
PRODUCT_IMAGE_MD5_STRING – контрольная сумма md5 передаваемого фото товара(строка)

DRY_RUN_STRING - тип запроса(boolean)
false - рабочий режим
true - тестовый режим

2.1.2.4 body(binary)

body(binary) содержит фото товара и представляет собой секцию multipart/form-data
со следующим заголовком

```
Content-Disposition: form-data; name="product_image"; filename="IMAGE_FILE_NAME"
Content-Type: image/jpeg
```

где `IMAGE_FILE_NAME` исходное имя файла снимка
Далее следует содержимое файла снимка (условно обозначение - `IMAGE_DATA`)

2.1.2.5 Пример запроса recognize

header

```
vision-protocol-version: 1.4
vision-token: 12345678-09A2-4348-AE91-7E1FA26E5E3A
```

body(text)

```
"session_id": "D49D284B-0A58-4FA5-847F-D2D0D49ECBB9"
"request_id": "D1430569-09A2-4348-AE91-7E1FA26E5E3A"
"product_image_md5": "08e511a927caf1f966b24b73ce53376d"
"dry_run": "false"
```

body(binary)

```
Content-Disposition: form-data; name="product_image"; filename="IMAGE_FILE_NAME"
Content-Type: image/jpeg
IMAGE_DATA
```

2.1.3 Ответ

2.1.3.1 Варианты ответов

Возможны следующие варианты ответов:
1) успешное выполнение
2) ошибка выполнения

2.1.3.2 Ответ - успешное выполнение

2.1.3.2.1 Ответ - успешное выполнение (общий формат)

http status:

```
HTTP_VERSION STATUS_CODE REASON_PHRASE
```

header:

```
vision-protocol-version: VISION_PROTOCOL_VERSION_UNQUOTED_STRING
```

body:

```
{
	"code": 1200,
	"name": "Success",
	"nn_version": NEURAL_NETWORK_VERSION_STRING,
	"result": PRODUCTS_JSON_ARRAY
	[
		PRODUCT_ITEM_JSON_STRUCT,
		PRODUCT_ITEM_JSON_STRUCT,
		PRODUCT_ITEM_JSON_STRUCT,
		PRODUCT_ITEM_JSON_STRUCT,
		PRODUCT_ITEM_JSON_STRUCT
	]
}
```

где

HTTP_VERSION - версия HTTP (HTTP/1.x)
STATUS_CODE - код HTTP
200: запрос обработан успешно
REASON_PHRASE - описание HTTP кода
OK
VISION_PROTOCOL_VERSION_UNQUOTED_STRING - использованная версия протокола
NEURAL_NETWORK_VERSION_STRING - Версия нейронной сети
PRODUCTS_JSON_ARRAY - массив топ-5 товаров
PRODUCT _ITEM_JSON _STRUCT - данные одного товара
"sku": PRODUCT_SKU_STRING - sku товара (строка)
"name": PRODUCT_NAME_STRING - название товара (строка)
"score": PRODUCT_CLASSIFICATION_SCORE_JSON_FLOAT - оценка классификации товара (чем выше значение, тем больше изображение соответствует данному товару)

2.1.3.2.2 Ответ - успешное выполнение (пример)

http status:

```
HTTP/1.1 200 OK
```

header:

```
vision-protocol-version: 1.4
```

body

```
{
	"code": 1200,
	"name": "Success",
	"nn_version": "10322f79c442fbec82307a8abf89d783",
	"result":
	 [
		{
			"name": "Картофель фасованный",
			"sku": "1036123",
			"score": 0.22121392800343484
		},
		{
			"name": "Гранат",
			"sku": "1036110",
			"score": 0.14674207251916002
		},
		{
			"name": "Яблоко Красное",
			"sku": "1036175",
			"score": 0.02716088717745504
		},
		{
			"name": "Лук репчатый",
			"sku": "1036133",
			"score": 0.017922750001051348
		},
		{
			"name": "Грейпфрут",
			"sku": "1036111",
			"score": 0.01113253679277089
		}
	]
}
```

2.1.3.3 Ответ - ошибка выполнения

2.1.3.3.1 Ответ - ошибка (общий формат)

http status:

```
HTTP_VERSION STATUS_CODE REASON_PHRASE
```

header:

```
vision-protocol-version: VISION_PROTOCOL_VERSION_UNQUOTED_STRING
```

body:

```
{
  "code": ERROR_CODE_NUMBER,
  "name": ERROR_NAME_STRING,
  "description": ERROR_DESCRIPTION_STRING
}
```

где

HTTP_VERSION - версия HTTP (HTTP/1.x)
STATUS_CODE - код HTTP
	400: логическая ошибка обработки запроса
	500: внутренняя ошибка сервера
REASON_PHRASE - описание HTTP кода
	Bad Request : если STATUS_CODE равен 400
	Internal Server Error : если STATUS_CODE равен 500
VISION_PROTOCOL_VERSION_UNQUOTED_STRING - использованная версия протокола
ERROR_CODE_NUMBER - код ошибки (число)
ERROR_NAME_STRING - краткое описание/название ошибки (строка)
ERROR_DESCRIPTION_STRING - подробное описание ошибки (строка)

2.1.3.3.2 Ответ - отсутствие поля в запросе

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1401,
  "name": "Missing Field Error",
  "description": "Missing Field Error. Missing ERROR_FIELD field in ERROR_BLOCK"
}
```

где

ERROR_FIELD - поле, с которым возникли проблемы

ERROR_BLOCK - `header` если ошибка возникла при проверке header или `body`, если ошибка возникла при проверке body запроса.

2.1.3.3.3 Ответ - ошибка разбора запроса

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1402,
  "name": "Request Parse Error",
  "description": "Request Parse Error. Corrupted ERROR_FIELD field in ERROR_BLOCK. Received ERROR_FIELD: ERROR_FIELD_ANY"
}
```

где

ERROR_FIELD - поле, с которым возникли проблемы

ERROR_FIELD_ANY - значение поля, с которым возникли проблемы

ERROR_BLOCK - `header` если ошибка возникла при проверке header или `body`, если ошибка возникла при проверке body

2.1.3.3.4 Ответ - ошибка: ошибка в маппинге

В случае, когда возникли ошибки в маппинге (используется несуществующий партнер и прочее):

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1403,
  "name": "Mapping Error",
  "description": "Mapping Error. Incorrect partner_id value. Partner PARTNER_ID does not exist"
}
```

где

PARTNER_ID - полученный идентификатор клиента(партнёра)

2.1.3.3.5 Ответ - ошибка сравнения хешей

В случае, когда возникла ошибка при сравнении полученного “product_image_md5” с посчитанным md5 на стороне сервера:

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1404,
  "name": "Hash Equality Error",
  "description": "Hash Equality Error. Received Md5 is not equal to calculated. Received: PRODUCT_IMAGE_MD5, calculated: CALCULATED_IMAGE_MD5"
}
```

где

PRODUCT_IMAGE_MD5 - полученный md5
CALCULATED_IMAGE_MD5 - посчитанный на стороне сервера md5

2.1.3.3.6 Ответ - ошибка в версии протокола

В случае, когда указана неправильная (недопустимая) версия протокола

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4.1.2.3.4a.b.
```

body:

```
{
  "code": 1405,
  "name": "Unsupported Protocol Version",
  "description": "Unsupported Protocol Version. Protocol Version: 1.4.1.2.3.4a.b. is not supported"
}
```

2.1.3.3.7 Ответ - ошибка с UUID

В случае, когда UUID поля неправильно заполнены

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1406,
  "name": "UUID Parse Error",
  "description": "UUID Parse Error. FIELD_NAME expected UUID like string. Received: FIELD_VALUE"
}
```

где

FIELD_NAME - название поля
FIELD_VALUE - значение поля

2.1.3.3.8 Ответ - ошибка: непредвиденная ошибка

http status:

```
HTTP/1.1 500 Internal Server Error
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1500,
  "name": "Internal Server Error",
  "description": "Unexpected Server Error"
}
```

2.1.3.3.9 Ответ - ошибка: ошибка в работе нейронной сети

http status:

```
HTTP/1.1 500 Internal Server Error
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1501,
  "name": "Neural Network Error",
  "description": " Neural Network Error. Neural network returns incorrect data"
}
```

2.2 Операция “upload”

2.2.1 Описание операции

Операция “upload” реализуется путём отправки на сервер POST запроса “upload” с последующим получением от сервера подтверждения о выполнении операции. Данный запрос хранит в себе:

1) изображение выбранного товара для сохранения на сервере;
2) атрибуты, описывающие выбранный товар;
3) опционально, информацию о наличии товаров в базе конечного устройства (структура "products_displayed");
4) дополнительные параметры, используемые сервером.

В зависимости от сценария запрос “upload” содержит следующие данные:
1) для сценария «Сбор данных»:
   информация о товаре, который был выбран покупателем
   не содержит в себе поле "products_displayed"
2) для сценария «Форсированный сбор данных»:
   информация о товаре, который был выбран для сбора датасета
   не содержит в себе поле "products_displayed"
3) для сценария «Распознавание»:
   информация о товаре, выбранном покупателем
   может содержать поле "products_displayed"
4) для сценария «Без распознавания»:
   информация о товаре, который был выбран покупателем
   не содержит в себе поле "products_displayed"

2.2.2 Запрос

2.2.2.1 Формат запроса

Возможные шаблоны URL запроса:
1) `https://`**`HOSTNAME`**`/upload`
2) `http://`**`IP_ADDRESS:PORT`**`/upload`

`IP_ADDRESS` – ip адрес сервера видео распознавания
`PORT` – TCP порт сервиса видео распознавания
`HOSTNAME` – хост сервиса видео распознавания

Запрос представляет собой http multipart/form-data контейнер, состоящий из 3 частей:
1) header
2) body(text)
3) body(binary)

2.2.2.2 header

header состоит из следующего набора текстовых полей

```
vision-protocol-version: VISION_PROTOCOL_VERSION_UNQUOTED_STRING
vision-token: VISION_TOKEN_UUID_UNQUOTED_STRING
```

VISION_PROTOCOL_VERSION_UNQUOTED_STRING - версия протокола
значение: 1.4

VISION_TOKEN_UUID_STRING - идентификатор конечного устройства,
значение: UUID без скобок

2.2.2.3 body(text)

body(text) состоит из следующего набора текстовых полей

```
"session_id": SESSION_ID_UUID_STRING
"request_id": REQUEST_ID_UUID_STRING
"partner_id": PARTNER_ID_STRING
"merchant_id": MERCHANT_ID_STRING
"product_image_md5": PRODUCT_IMAGE_MD5_STRING
"dry_run": DRY_RUN_BOOLEAN_STRING
"forced_dataset_collection": FORCED_DATASET_BOOLEAN_STRING
"selection_info": SELECTION_INFO_JSON_STRUCT
{
      "product": SELECTION_INFO_PRODUCT_JSON_STRUCT
      {
            "sku": PRODUCT_SKU_STRING,
            "name": PRODUCT_NAME_STRING,
            "non_weight": PRODUCT_IS_NON_WEIGHT_JSON_BOOLEAN,
            "item_quantity": PRODUCT_ITEM_QUANTITY_JSON_NUMBER
      },
      "selection_type": PRODUCT_SELECTION_TYPE_NUMBER,
      "local_time": PRODUCT_SELECTION_LOCAL_TIME_STRING
}
"products_displayed": PRODUCTS_DISPLAYED_JSON_STRUCT
{
    "products": PRODUCTS_JSON_ARRAY
    [
      PRODUCT_ITEM_JSON_STRUCT,
      PRODUCT_ITEM_JSON_STRUCT,
      PRODUCT_ITEM_JSON_STRUCT,
      PRODUCT_ITEM_JSON_STRUCT,
      PRODUCT_ITEM_JSON_STRUCT
    ]
}
```

Описание полей

SESSION_ID_UUID_STRING - идентификатор сеанса
значение: UUID без скобок

REQUEST_ID_UUID_STRING - идентификатор запроса
значение: UUID без скобок

PARTNER_ID_STRING - идентификатор клиента(партнёра)

MERCHANT_ID_STRING - идентификатор магазина

PRODUCT_IMAGE_MD5_STRING - md5 передаваемого изображения

DRY_RUN_BOOLEAN_STRING - тип запроса
- false - рабочий режим
- true - тестовый режим

FORCED_DATASET_BOOLEAN_STRING - признак форсированного сбора датасета
- false - сбор данных (обычная работа)
- true - форсированный сбор данных

SELECTION_INFO_JSON_STRUCT - структура с описанием выбора покупателем

SELECTION_INFO_PRODUCT_JSON_STRUCT - описание товара, выбранного покупателем
- PRODUCT_SKU_STRING - sku товара
- PRODUCT_NAME_STRING - название товара
- PRODUCT_IS_NON_WEIGHT_BOOLEAN - тип товара (штучный/весовой)
  - true - штучный(невесовой) товар
  - false - весовой товар
- PRODUCT_ITEM_QUANTITY_JSON_NUMBER - «количество» товара:
  - для штучного товара - количество (в штуках)
  - для весового товара - вес (в килограммах)

PRODUCT_SELECTION_TYPE_NUMBER - тип выбора покупателя
0 - зарезервировано (товар не выбран)
1 - до взвешивания по коду
2 - до взвешивания из популярных
3 - до взвешивания по наименованию
4 - до взвешивания из группы
5 - после взвешивания по коду
6 - после взвешивания из популярных
7 - после взвешивания по наименованию
8 - после взвешивания из группы
9 - после взвешивания из результатов распознавания

PRODUCT_SELECTION_LOCAL_TIME_STRING - локальное время выбора товара в формате
"YYYY-MM-DD hh:mm:ss.zzz"
YYYY - год
MM - месяц 01-12
DD - дата месяца 01-31
hh - часы 00-23
mm - минуты 00-59
ss - секунды 00-59
zzz – миллисекунды 000-999

PRODUCTS_DISPLAYED_JSON_STRUCT - опциональная структура, показывающая наличие в БД клиентского устройства товаров, полученных от сервера в результате выполнения операции recognize. Данная структура при необходимости может использоваться для корректировки БД товаров на сервере с учётом их наличия в БД клиентского устройств.

PRODUCTS_JSON_ARRAY - массив товаров показанных покупателю для выбора
на основе списка, полученного в ответ на запрос recognize

PRODUCT_JSON_ITEM_STRUCT - данные одного показанного товара
"sku": PRODUCT_SKU_STRING - sku товара (строка)
"name": PRODUCT_NAME_STRING - название товара (строка)
"found": PRODUCT_FOUND_IN_LOCAL_DB_JSON_BOOLEAN - признак наличия товара в БД конечного устройства
PRODUCT_FOUND_IN_LOCAL_DB_BOOLEAN
false - товара нет в БД конечного устройства
true - товар есть в БД конечного устройства

2.2.2.4 body(binary)

body(binary) содержит фото товара и представляет собой секцию multipart/form-data
со следующим заголовком

```
Content-Disposition: form-data; name="product_image"; filename="IMAGE_FILE_NAME"
Content-Type: image/jpeg
```

где IMAGE_FILE_NAME исходное имя файла снимка
Далее следует содержимое файла снимка (условно обозначение - IMAGE_DATA)

2.2.2.5 Пример запроса (products_displayed)

header

```
vision-protocol-version: 1.4
vision-token: 12345678-09A2-4348-AE91-7E1FA26E5E3A
```

body(text)

```
"session_id": "D49D284B-0A58-4FA5-847F-D2D0D49ECBB9"
"request_id": "D1430569-09A2-4348-AE91-7E1FA26E5E3A"
"merchant_id": "338991"
"partner_id": "349"
"product_image_md5": "08e511a927caf1f966b24b73ce53376d"
"dry_run": "false"
"forced_dataset_collection": "false"
"selection_info":
{
      "product":
      {
            "sku": "288911000000",
            "name": "Грейпфрут",
            "non_weight": false,
            "item_quantity": 1.234
      },
      "selection_type": 5,
      "local_time": "2021-10-07 12:36:56.214"
}
"products_displayed":
{
  "products":
  [
    {
      "sku": "288911000000",
      "name": "Грейпфрут",
      "found": true
    },
    {
      "sku": "288902000000",
      "name": "Апельсин",
      "found": true
    },
    {
      "sku": "240296900000",
      "name": "Лук репчатый",
      "found": true
    },
    {
      "sku": "281225100000",
      "name": "Гранат",
      "found": false
    },
    {
      "sku": "221436600000",
      "name": "Манго",
      "found": false
    }
  ]
}
```

body(binary)

```
Content-Disposition: form-data; name="product_image"; filename="IMAGE_FILE_NAME"
Content-Type: image/jpeg
IMAGE_DATA
```

2.2.2.6 Пример запроса (без products_displayed)

header
```
vision-protocol-version: 1.4
vision-token: 12345678-09A2-4348-AE91-7E1FA26E5E3A
```

body(text)
```
"session_id": "D49D284B-0A58-4FA5-847F-D2D0D49ECBB9"
"request_id": "D1430569-09A2-4348-AE91-7E1FA26E5E3A"
"merchant_id": "338991"
"partner_id": "349"
"product_image_md5": "08e511a927caf1f966b24b73ce53376d"
"dry_run": "false"
"forced_dataset_collection": "false"
"selection_info":
{
      "product":
      {
            "sku": "288911000000",
            "name": "Грейпфрут",
            "non_weight": false,
            "item_quantity": 1.234
      },
      "selection_type": 5,
      "local_time": "2021-10-07 12:36:56.214"
}
```

body(binary)
```
Content-Disposition: form-data; name="product_image"; filename="IMAGE_FILE_NAME"
Content-Type: image/jpeg
IMAGE_DATA
```

2.2.3 Ответ

2.2.3.1 Варианты ответов

Возможны следующие варианты ответов
1) Ответ при успешном выполнении
2) Ответ при ошибке выполнения

2.2.3.2 Ответ - успешное выполнение

2.2.3.2.1 Ответ - успешное выполнение (общий формат)

В случае успешного выполнения сервер должен вернуть ответ в следующем формате:
http status:
```
      HTTP_VERSION STATUS_CODE REASON_PHRASE
```
header:
```
      vision-protocol-version: VISION_PROTOCOL_VERSION_UNQUOTED_STRING
```
body:
```
{
  "code": 1200,
  "name": "Success"
}
```
где
- HTTP_VERSION - версия HTTP (HTTP/1.x)
- STATUS_CODE - код HTTP
  - 200: запрос обработан успешно
- REASON_PHRASE - описание HTTP кода
  - OK
- VISION_PROTOCOL_VERSION_UNQUOTED_STRING - использованная версия протокола

2.2.3.2.2 Ответ - успешное выполнение (пример)

В случае успешного выполнения сервер должен вернуть ответ в следующем формате:
http status:

```
HTTP/1.1 200 OK
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1200,
  "name": "Success"
}
```

2.2.3.3 Ответ - ошибка выполнения

2.2.3.3.1 Ответ - ошибка (общий формат)

http status:

```
HTTP_VERSION STATUS_CODE REASON_PHRASE
```

header:

```
vision-protocol-version: VISION_PROTOCOL_VERSION_UNQUOTED_STRING
```

body:

```
{
  "code": ERROR_CODE_NUMBER,
  "name": ERROR_NAME_STRING,
  "description": ERROR_DESCRIPTION_STRING
}
```

где

HTTP_VERSION - версия HTTP (HTTP/1.x)
STATUS_CODE - код HTTP
400: логическая ошибка обработки запроса
500: внутренняя ошибка сервера
REASON_PHRASE - описание HTTP кода
Bad Request : если STATUS_CODE равен 400
Internal Server Error : если STATUS_CODE равен 500
VISION_PROTOCOL_VERSION_UNQUOTED_STRING - использованная версия протокола
ERROR_CODE_NUMBER - код ошибки (число)
ERROR_NAME_STRING - краткое описание/название ошибки (строка)
ERROR_DESCRIPTION_STRING - подробное описание ошибки (строка)

2.2.3.3.2 Ответ - отсутствие поля в запросе

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1401,
  "name": "Missing Field Error",
  "description": "Missing Field Error. Missing ERROR_FIELD field in ERROR_BLOCK"
}
```

где

ERROR_FIELD - поле, с которым возникли проблемы
ERROR_BLOCK - `header` если ошибка возникла при проверке header или `body`, если ошибка возникла при проверке body

2.2.3.3.3 Ответ - ошибка разбора запроса

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1402,
  "name": "Request Parse Error",
  "description": "Request Parse Error. Corrupted ERROR_FIELD field in ERROR_BLOCK. Received ERROR_FIELD: ERROR_FIELD_ANY"
}
```

где

ERROR_FIELD - поле, с которым возникли проблемы

ERROR_FIELD_ANY - значение поля, с которым возникли проблемы

ERROR_BLOCK - `header` если ошибка возникла при проверке header или `body`, если ошибка возникла при проверке body

2.2.3.3.4 Ответ - ошибка в маппинге

В случае, когда возникли ошибки в маппинге (используется несуществующий партнер и прочее):

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1403,
  "name": "Mapping Error",
  "description": "Mapping Error. Incorrect partner_id value. Partner PARTNER_ID  does not exist"
}
```

где

PARTNER_ID - полученный идентификатор клиента(партнёра)

2.2.3.3.5 Ответ - ошибка сравнения хешей

В случае, когда возникла ошибка при сравнении полученного “product_image_md5” с посчитанным md5 на стороне сервера:

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1404,
  "name": "Hash Equality Error",
  "description": "Hash Equality Error. Received Md5 is not equal to calculated. Received: PRODUCT_IMAGE_MD5, calculated: CALCULATED_IMAGE_MD5"
}
```

где

PRODUCT_IMAGE_MD5 - полученный md5

CALCULATED_IMAGE_MD5 - посчитанный на стороне сервера md5

2.2.3.3.6 Ответ - ошибка в версии протокола

В случае, когда указана неправильная (недопустимая) версия протокола

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4.1.2.3.4a.b.
```

body:

```
{
  "code": 1405,
  "name": "Unsupported Protocol Version",
  "description": "Unsupported Protocol Version. Protocol Version: 1.4.1.2.3.4a.b. is
not supported"
}
```

2.2.3.3.7 Ответ - ошибка с UUID

В случае, когда UUID поля неправильно заполнены

http status:

```
HTTP/1.1 400 Bad Request
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1406,
  "name": "UUID Parse Error",
  "description": "UUID Parse Error. FIELD_NAME expected UUID like string. Received:
FIELD_VALUE"
}
```

где

FIELD_NAME - название поля
FIELD_VALUE - значение поля

2.2.3.3.8 Ответ - непредвиденная ошибка

http status:

```
HTTP/1.1 500 Internal Server Error
```

header:

```
vision-protocol-version: 1.4
```

body:

```
{
  "code": 1500,
  "name": "Internal Server Error",
  "description": "Unexpected Server Error"
}
```

3. Список расширенных кодов ошибок

```
code: 1200
name: Success
http code: 200
```
Описание: Код возникает при штатной работе всех сервисов

```
code: 1401
name: Missing Field Error
http code: 400
```
Описание: Код возникает при отсутствии необходимых полей в header или в body

```
code: 1402
name: Request Parse Error
http code: 400
```
Описание: Код возникает при ошибке заполнения какого-либо поля в header или в body

```
code: 1403
name: Mapping Error
http code: 400
```
Описание: Код возникает при ошибке в указании партнера (также ошибка может возникнуть при неправильной загрузке базы маппинга на сервер видео распознавания)

```
code: 1404
name: Hash Equality Error
http code: 400
```
Описание: Код возникает при несоответствии полученного хеша (md5) с посчитанным на стороне сервера

```
code: 1405
name: Unsupported Protocol Version
http code: 400
```
Описание: Код возникает при использовании неподдерживаемой версии протокола

```
code: 1406
name: UUID Parse Error
http code: 400
```
Описание: Код возникает при неправильном заполнении UUID полей (не соответствует типу UUID)

```
code: 1500
name: Internal Server Error
http code: 500
```
Описание: Код возникает при непредвиденной ошибке на стороне сервера

```
code: 1501
name: Neural Network Error
http code: 500
```
Описание: Код возникает при ошибке работы нейросети