



Бастион-3 – Web API. Руководство программиста gRPC

Версия 2026.1

(17 апреля, 2026)



Самара, 2026

Содержание

1.	Общие сведения.....	11
2.	Условия применения	12
3.	Настройка	13
4.	Подключение к gRPC Web API	14
5.	Взаимодействие с сервером системы через gRPC	15
5.1.	Примеры кода клиента	15
5.2.	AuthorizationService	15
5.2.1.	Метод Login	15
5.2.2.	Метод Logout.....	16
5.2.3.	Метод RefreshToken	16
5.2.4.	Метод Unauthorized	17
5.3.	ServerInfoService	17
5.3.1.	Метод GetModules	17
5.3.2.	Метод GetServerState	18
5.4.	UpdateDataService.....	18
5.4.1.	Метод UpdateData	18
5.4.1.1.	Редактирование уровней доступа	20
5.4.1.2.	Редактирование карт доступа	21
5.4.1.3.	Редактирование словарей и их значений.....	21
5.4.1.4.	Редактирование пропусков	23
5.4.1.5.	Редактирование персон.....	23
5.4.1.6.	Редактирование временных блоков.....	24
5.4.1.7.	Редактирование дополнительных полей устройств	25
5.4.1.8.	Редактирование организационной структуры.....	25
5.4.1.9.	Редактирование транспортных средств и их статусов.....	27
5.4.1.10.	Редактирование операторов, ролей операторов и полномочий.....	28
5.4.1.11.	Редактирование рабочих дней.....	29
5.4.1.12.	Редактирование графиков сменности и позиций графиков сменности	30
5.4.1.13.	Редактирование трудовых договоров	31
5.4.1.14.	Редактирование дополнительных полей пропусков.....	32
5.4.1.15.	Редактирование привязки устройств к территориям.....	32
5.4.1.16.	Редактирование биометрических шаблонов	33

5.4.1.17.	Редактирование настроек филиала репликации и настроек репликации пропусков	33
5.5.	AccessLevelService	34
5.5.1.	Метод GetAccessLevels	34
5.5.2.	Метод GetAccessLevel	35
5.5.3.	Метод GetAccessLevelContents	36
5.5.4.	Метод GetAccessLevelContentCompositions	37
5.5.5.	Метод AccessLevelChanged	38
5.5.6.	Метод AccessLevelContentChanged	38
5.6.	CardService	38
5.6.1.	Метод GetCard	38
5.6.2.	Метод GetCards	40
5.6.3.	Метод ReturnCardToNormalState	43
5.6.4.	Метод CardChanged	43
5.7.	DriverPassProfileService	43
5.7.1.	Метод GetDriverPassProfileTypes	43
5.7.2.	Метод GetDriverPassProfiles	44
5.7.3.	Метод GetDriverPassProfileForPass	45
5.8.	ExternalIntegration	46
5.8.1.	Метод ExternalAccessConfirmRequired	46
5.8.2.	Метод ConfirmAccess	47
5.8.3.	Метод DenyAccess	47
5.9.	OrganizationStructureService	47
5.9.1.	Метод GetOrganizationStructureNodes	47
5.9.2.	Метод OrganizationStructureChanged	48
5.10.	ControlAreaService	48
5.10.1.	Метод GetControlArea	49
5.10.2.	Метод GetControlAreas	49
5.10.3.	Метод GetControlAreaPassPointLinks	50
5.10.4.	Метод ControlAreaChanged	50
5.10.5.	Метод ControlAreaPassPointLinkUpdated	51
5.11.	DictionariesService	51
5.11.1.	Метод GetDictionaryHeaders	51
5.11.2.	Метод GetDictionaryRecords	52
5.11.3.	Метод GetDictionaryRecord	53
5.11.4.	Метод DictionaryRecordChanged	53



5.11.5.	Метод GetDictionaryHeader	54
5.11.6.	Метод DictionaryHeaderChanged.....	55
5.12.	PassService	55
5.12.1.	Метод GetPass.....	55
5.12.2.	Метод GetPasses.....	57
5.12.3.	Метод GetPassesForPerson	58
5.12.4.	Метод GetPassPinCode	58
5.12.5.	Метод GetPassPinCodes	59
5.12.6.	Метод GetCardReplacementInfo	60
5.12.7.	Метод GetPassSecurityControlGroupSettings.....	61
5.12.8.	Метод IssuePass	61
5.12.9.	Метод ReturnPass	62
5.12.10.	Метод WithdrawPass	62
5.12.11.	Метод ProlongPass	63
5.12.12.	Метод ReplaceCardWithReturn.....	63
5.12.13.	Метод ReplaceCardWithWithdraw.....	64
5.12.14.	Метод PassChanged	64
5.13.	PassCategoryService	64
5.13.1.	Метод GetPassCategory	65
5.13.2.	Метод GetPassCategories	66
5.13.3.	Метод GetPassCategoriesAvailabilityInfo	67
5.13.4.	Метод GetOrganizationAllowedPassCategories	68
5.13.5.	Метод PassCategoryChanged	68
5.14.	PersonService.....	68
5.14.1.	Метод GetPerson	69
5.14.2.	Метод GetPersonAdditionalFields	70
5.14.3.	Метод GetPersonAllInfo	72
5.14.4.	Метод GetPersonAllInfos	74
5.14.5.	Метод GetPersonPhoto	75
5.14.6.	Метод GetPersonPhotoHash	76
5.14.7.	Метод GetPersonPhotoThumbnail	76
5.14.8.	Метод GetPersons	77
5.14.9.	Метод PersonAdditionalFieldsChanged.....	79
5.14.10.	Метод PersonChanged.....	79
5.14.11.	Метод PersonInfoChanged	79



5.14.12.	Метод PersonPhotoChanged.....	80
5.15.	SearchPassService.....	80
5.15.1.	SearchPasses.....	80
5.15.2.	Фильтры пропусков.....	81
5.16.	PersonLocationService.....	84
5.16.1.	Метод GetPersonLocationCounters.....	84
5.16.2.	Метод GetPersonLocations.....	85
5.16.3.	Метод PersonLocationChanged.....	86
5.16.4.	Метод PersonLocationCounterChanged.....	87
5.17.	SecurityControlGroupService.....	87
5.17.1.	Метод GetSecurityControlGroups.....	88
5.17.2.	Метод GetSecurityControlGroup.....	88
5.17.3.	Метод GetSecurityControlGroupContent.....	89
5.18.	StopListService.....	90
5.18.1.	Метод GetBlockedPersons.....	90
5.18.2.	Метод GetBlockedPersonByPersonId.....	91
5.18.3.	Метод AddPersonToStopList.....	91
5.18.4.	Метод RemovePersonFromStopList.....	92
5.18.5.	Метод BlockedPersonChanged.....	92
5.19.	TimeBlockService.....	92
5.19.1.	Метод GetTimeBlock.....	93
5.19.2.	Метод GetTimeBlocks.....	94
5.19.3.	Метод TimeBlockChanged.....	95
5.20.	AdditionalFieldService.....	95
5.20.1.	GetAdditionalFieldDescriptor.....	96
5.20.2.	GetAdditionalFieldDescriptors.....	96
5.20.3.	GetAdditionalFieldValues.....	97
5.20.4.	GetAdditionalFieldValueChanged.....	98
5.21.	DriverInfoService.....	99
5.21.1.	Глоссарий.....	99
5.21.1.1.	Тип драйвера.....	99
5.21.1.2.	Базовый тип устройства.....	99
5.21.1.3.	Сервер (хост) оборудования.....	99
5.21.1.4.	Экземпляр (инстанс) драйвера.....	99
5.21.1.5.	Устройство.....	100

5.21.1.6.	Тип события устройства	100
5.21.1.7.	Тип команды_действия устройства	100
5.21.1.8.	Тип группы устройств	101
5.21.2.	Метод GetDriverHosts	101
5.21.3.	Метод GetDriverTypes	101
5.21.4.	Метод GetDriverInstances	102
5.21.5.	Метод GetDriverInstanceDevice	104
5.21.6.	Метод GetDevice	105
5.21.7.	Метод GetDriverActionTypes	106
5.21.8.	Метод GetDriverMessageTypes	106
5.21.9.	Метод ExecuteDeviceCommand	108
5.21.10.	Метод GetDeviceState	108
5.21.11.	Метод ConfigurationChanged	109
5.21.12.	Метод GetDeviceGroupTypes	110
5.21.13.	Метод GetDeviceGroupMembers	110
5.21.14.	Метод GetDeviceGroupOwners	111
5.21.15.	Метод GetReaderToPassPointLinks	111
5.21.16.	Метод GetMessageTypes	112
5.22.	SearchDevicesService	113
5.22.1.	SearchDevices	114
5.22.2.	Фильтры устройств	114
5.23.	MessageInfoService	115
5.23.1.	Глоссарий1	115
5.23.1.1.	Профиль события	115
5.23.2.	Метод GetMessageProfile	115
5.23.3.	Метод GetMessageProfiles	116
5.23.4.	Метод GetUnconfirmedMessages	117
5.23.5.	Метод ConfirmMessages	118
5.23.6.	Метод MessageProfileChanged	119
5.23.7.	Метод NewMessageProcessed	119
5.23.8.	Метод MessagesConfirmed	120
5.23.9.	Метод GenerateDeviceMessages	120
5.24.	ProtocolInfoService	121
5.24.1.	Метод GetMessages	121
5.24.1.1.	Фильтры событий	125

5.24.1.2.	Дополнительная информация	128
5.24.2.	Метод GetLastProtocolMessageInfo	134
5.25.	AuditService.....	134
5.25.1.	Метод GetAuditEntries	135
5.25.1.1.	Фильтры записей журнала аудита.....	137
5.26.	TransportService	138
5.26.1.	Метод GetTransportStatuses.....	138
5.26.2.	Метод GetTransports.....	139
5.26.3.	Метод GetTransport	140
5.26.4.	Метод GetTransportPhoto	141
5.26.5.	Метод GetTransportPhotoHash.....	142
5.26.6.	Метод GetTransportPhotoThumbnail	142
5.26.7.	Метод TransportChanged	143
5.26.8.	Метод TransportPhotoChanged	143
5.26.9.	Метод TransportStatusChanged	144
5.27.	OperatorsInfoService	144
5.27.1.	Метод GetRoles.....	144
5.27.2.	Метод GetOperators	145
5.27.3.	Метод GetOperator	146
5.27.4.	Метод GetRole.....	147
5.27.5.	Метод GetOperatorInfos.....	148
5.27.6.	Метод GetOperatorInfo.....	148
5.27.7.	Метод ChangePassword	149
5.27.8.	Метод RoleChanged	150
5.27.9.	Метод OperatorChanged	150
5.27.10.	Метод GetGeneralDevicePermissions.....	150
5.27.11.	Метод GetRestrictedDeviceActions.....	151
5.27.12.	Метод GeneralDevicePermissionsChanged.....	152
5.27.13.	Метод RestrictedDeviceActionsChanged.....	152
5.27.14.	Метод GetAllBastionSettings	153
5.27.15.	Метод GetRoleBastionSettings	154
5.27.16.	Метод GetApplicationPermissions	155
5.27.17.	Метод CheckDevicePermission	155
5.27.18.	Метод CheckApplicationPermission	156
5.27.19.	Метод ApplicationPermissionsChanged.....	156



5.27.20.	Метод RoleBastionSettingsChanged.....	157
5.28.	RegimeService.....	157
5.28.1.	Метод GetWorkDayTypes.....	157
5.28.2.	Метод GetRegimes	158
5.28.3.	Метод RegimeChanged	159
5.29.	ShiftScheduleService	160
5.29.1.	Метод GetShiftSchedules.....	160
5.29.2.	Метод GetShiftScheduleItems	161
5.29.3.	Метод GetShiftScheduleItemsForShiftSchedule	161
5.29.4.	Метод ShiftScheduleChanged	162
5.29.5.	Метод ShiftScheduleItemChanged.....	162
5.29.6.	Метод GetShiftScheduleOrganizations	163
5.30.	WorkContractService	163
5.30.1.	Метод GetWorkContract.....	163
5.30.2.	Метод GetWorkContractsForPerson	164
5.30.3.	Метод GetPersonCountWithoutWorkContract.....	165
5.30.4.	Метод WorkContractsCreate	166
5.30.5.	Метод WorkContractChanged	166
5.30.6.	Метод GetCalcMethods.....	166
5.30.7.	Метод GetShiftScheduleId	167
5.30.8.	Метод GetAllPassCategoryAtdSettings	167
5.31.	PassAdditionalFieldService.....	168
5.31.1.	Метод GetPassAdditionalFieldDescriptor	168
5.31.2.	Метод GetPassAdditionalFieldDescriptors	168
5.31.3.	Метод GetPassAdditionalFieldValues	169
5.31.4.	Метод GetPassCategoryTypedAdditionalFields	170
5.31.5.	Метод PassAdditionalFieldDescriptorChanged.....	171
5.31.6.	Метод PassAdditionalFieldValueChanged.....	171
5.32.	DeviceControlAreaLinkService	172
5.32.1.	Метод GetDeviceControlAreaLinks.....	172
5.32.2.	Метод GetLinkedDevicesByControlAreaId	173
5.32.3.	Метод GetLinkedControlAreaTo	174
5.33.	BiometryService	174
5.33.1.	Метод GetBioTemplateTypes	174
5.33.2.	Метод GetPersonBioTemplateInfos	175



5.33.3.	Метод GetPersonBioTemplates.....	176
5.33.4.	Метод GetPersonsBioTemplates.....	177
5.33.5.	Метод PersonBioTemplatesChanged	177
5.34.	ReplicationBranchService.....	178
5.34.1.	Коды типов сущностей репликации.....	178
5.34.2.	Метод ApplyBranchChange.....	179
5.34.3.	Метод ApplyCenterChange	181
5.34.4.	Метод AvoidBranchChange.....	184
5.34.5.	Метод AvoidCenterChange	184
5.34.6.	Метод BranchApplyMessageReceived.....	185
5.34.7.	Метод BranchChangeStateChanged	185
5.34.8.	Метод BranchListChanged.....	186
5.34.9.	Метод CardOwnerChanged	186
5.34.10.	Метод CenterChangeStateChanged	186
5.34.11.	Метод ClearReplicationProtocol	187
5.34.12.	Метод GetAllCardsReplicationInfo	187
5.34.13.	Метод GetAllOrganizationsReplicationInfo.....	188
5.34.14.	Метод GetAllPassCategoryOrganizationReplicationProperties	188
5.34.15.	Метод GetAllPassesReplicationInfo.....	189
5.34.16.	Метод GetAllPersonsReplicationInfo	189
5.34.17.	Метод GetBranchChangeApplyInfo	190
5.34.18.	Метод GetBranchChanges	191
5.34.19.	Метод GetBranches	192
5.34.20.	Метод GetBranchFilteredHistory	193
5.34.21.	Метод GetBranchFilteredHistoryDateRange.....	197
5.34.22.	Метод GetBranchFilteredHistorySize	198
5.34.23.	Метод GetCenterChanges	198
5.34.24.	Метод GetCenterFilteredHistory	200
5.34.25.	Метод GetCenterFilteredHistoryDateRange.....	202
5.34.26.	Метод GetCenterFilteredHistorySize	203
5.34.27.	Метод GetClearReplicationProtocolScheduleSettings.....	204
5.34.28.	Метод GetDefaultConflictResolveRules	205
5.34.29.	Метод GetForeignDataEditSettings	206
5.34.30.	Метод GetGlobalAccessLevels	207
5.34.31.	Метод GetIsInReplication	207



5.34.32.	Метод GetPassCategoryOrganizationReplicationProperties.....	208
5.34.33.	Метод GetPassesOwnerUids	208
5.34.34.	Метод GetPassReplicationProperties	209
5.34.35.	Метод GetReplicationMetrics	210
5.34.36.	Метод GetReplicationScheduleSettings.....	211
5.34.37.	Метод IsInReplicationChanged	212
5.34.38.	Метод NewBranchChangeCreated	212
5.34.39.	Метод NewCenterChangesReceived	212
5.34.40.	Метод NextClearReplicationProtocolDateChanged	212
5.34.41.	Метод NextReplicationDateChanged	212
5.34.42.	Метод OrganizationOwnerChanged	213
5.34.43.	Метод PassOwnerChanged.....	213
5.34.44.	Метод PersonOwnerChanged	213
5.34.45.	Метод RefreshCenterChanges	214
5.34.46.	Метод ReplicationProtocolCleaned	214
6.	Приложения	215
6.1.	Приложение 1. Тип google.protobuf.Any.....	215
6.2.	Приложение 2. Ошибки, возвращаемые ПК «Бастиян-3».....	215
6.3.	Приложение 3. Основные элементы фильтрации	216
6.4.	Приложение 4. Состояния устройств	219

1. Общие сведения

Модуль «Бастион-3 – gRPC» предназначен для организации информационного взаимодействия внешних систем с ПК «Бастион-3» через gRPC-протокол. Подробнее о самом протоколе можно прочитать в [официальной документации](#)¹.

Ключевые сценарии использования, для которых может применяться API, включают:

1. Взаимодействие с системами учёта посетителей, кадровыми и бухгалтерскими системами – создание, изменение персональных, материальных и транспортных пропусков в ПК «Бастион-3». Выполнение операций с пропусками (выдача, блокировка / разблокировка, продление, возврат, изъятие). Получение дерева подразделений, работа с подразделениями. Получение списка точек прохода и территорий. Получение событий по запросу за период времени. Получение списка уровней доступа. Получение событий УРВ.
2. Получение событий и передача команд ПК «Бастион-3» в реальном времени. Может использоваться для разработки дополнительных АРМ-ов, для интеграции со сторонними системами, когда не требуется использование стандартных протоколов OPC UA или SNMP.

1. <https://grpc.io/>

2. Условия применения

На "Бастион-3 – Web API" при использовании gRPC распространяются те же требования к аппаратной и программной платформе, что и для ПК «Бастион-3».

Для работы с gRPC API требуется лицензия «Бастион-3 – Web API»..

Для работы требуется ПК «Бастион-3» версии не ниже 2024.1.

3. Настройка

По умолчанию Web API ПК «Бастион-3» отключен. Для его включения и настройки необходимо воспользоваться утилитой «Локальные параметры» из комплекта поставки ПК «Бастион-3» или консольной утилитой BCnfg.

Точка подключения нужного типа добавляется по кнопке "Добавить прослушиватель" [Рис 1](#).

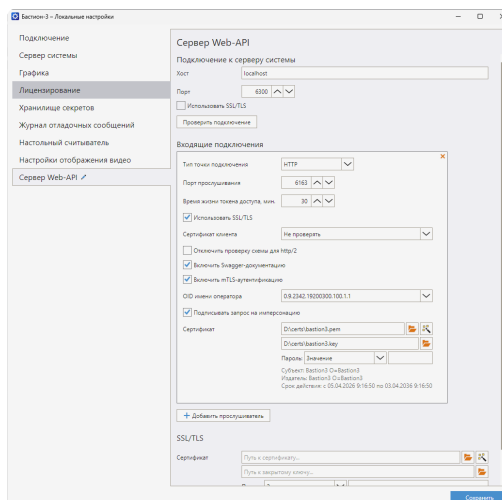


Рис. 1. Настройки подключения HTTP API

Включить Swagger-документацию - включение автоматической документации.

Включить mTLS-аутентификацию - Включение mTLS-аутентификации.

OID имени оператора - OID для определения логина оператора.



Внимание!

При аутентификации с помощью сертификата имя оператора, используемое для входа в систему, берётся из значения компонента имени субъекта (владельца) сертификата. Данный параметр задаёт OID, которым определяется какой именно компонент нужно использовать 0.9.2342.19200300.100.1.1 (userID, UID) или 2.5.4.3 (commonName, CN).

Подписывать запрос на имперсонацию - Включение подписи запроса делегирования полномочий.

Сертификат - путь к файлу SSL-сертификата, к файлу закрытого ключа сертификата для подписи запроса на делегирование полномочий и пароль для закрытого ключа со способом его получения. В поле необходимо указать путь к файлу с сертификатом, либо к файлу в формате PKCS#12, содержащему сертификат и закрытый ключ к нему. Сертификат должен быть сохранён в PEM-формате. Для закрытого ключа поддерживаются форматы PKCS#1 и PKCS#8 (предпочтителен PKCS#8).

Подробнее см. разделы "Сервер Web API" и "Настройка сервера Web API" в документе «Бастион-3. Руководство администратора».

4. Подключение к gRPC Web API

Подключение выполняется по адресу `grpc://{server_address}:{port}`, где `{server_address}` - адрес выделенного Web API сервера, `{port}` – порт для подключений посредством gRPC, выбранный в локальных настройках.

5. Взаимодействие с сервером системы через gRPC

В этом разделе приведены описание и примеры использования сервисов, доступных через gRPC Web API.

5.1. Примеры кода клиента

На сайте [GitVerse](https://gitverse.ru/)² доступны [Репозитории ТвинПро](https://gitverse.ru/twinpro)³, содержащие исходные коды примеров клиентских утилит для работы с gRPC Web API ПК «Бастион-3»:

- клиентская утилита на языке с#: <https://gitverse.ru/twinpro/bastion3-grpc-client-sample-dotnet>;
- клиентская утилита на языке Java: <https://gitverse.ru/twinpro/bastion3-grpc-client-sample-java>.

5.2. AuthorizationService

Описание сервиса приведено в файле authorization.proto. Данный сервис используется при авторизации в системе. Для выполнения почти всех запросов ко всем сервисам, доступным через gRPC, необходимо пройти авторизацию и получить токен доступа.

5.2.1. Метод Login

Метод используется для непосредственной авторизации в системе и получения токена доступа. Для авторизации необходимо в параметрах запроса (LoginRequest) передать параметры аутентификации (поле AuthenticationParameters). Сейчас доступно два типа параметров:

Пара логин/пароль. Пример тела запроса:

```
{
  "UserAndPassword": {
    "user": "q",
    "password": "q"
  }
}
```

Секретное слово. Пример тела запроса:

```
{
  "Otp": {
    "user": "q",
    "secret_word": "secret"
  }
}
```

Клиентский сертификат. Пример тела запроса:

2. <https://gitverse.ru/>

3. <https://gitverse.ru/twinpro>

```
{
  "ClientCertificate": {}
}
```

При успешной авторизации будет возвращено:

```
{
  "session_id": 0,
  "access_token": "",
  "access_token_expire_time": {
    "seconds": 0,
    "nanos": 0
  }
}
```

Сообщение состоит из:

- `session_id` – уникальный идентификатор вашей сессии;
- `access_token` – токен доступа, используемый для дальнейших обращений к серверу системы от имени авторизованного пользователя. Токен доступа при последующем выполнении других запросов передаётся в заголовке `authorization` в формате: *Bearer <токен доступа>*. Токен действителен только во время текущей сессии, если сервер системы был перезагружен, то токен перестает быть действительным;
- `access_token_expire_time` – время окончания действия токена доступа.

5.2.2. Метод Logout

Запрос на завершение сессии ранее авторизованного пользователя. Выполняется с пустым телом запроса, в качестве успешного результата ответ приходит с пустым телом. Запрос может быть выполнен только с использованием токена доступа.

5.2.3. Метод RefreshToken

Запрос на обновление токена доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа сервера системы:

```
{
  "access_token": "",
  "access_token_expire_time": {
    "seconds": 0,
    "nanos": 0
  }
}
```

- `access_token` – новый токен доступа;
- `access_token_expire_time` – время окончания действия нового токена доступа.

После выполнения запроса, старый токен доступа, будет не действителен. Для дальнейших обращений необходимо использовать новый токен доступа, полученный в качестве ответа на запрос.

5.2.4. Метод `Unauthorized`

Запрос на отслеживание активной сессии. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Если сессия завершилась в результате действия пользователя (например, в результате завершения сессии пользователя путем отправки запроса `Logout`), то получаем следующий ответ:

```
{
  "Normal": {}
}
```

Если завершение сессии вызвано ошибкой, то ответ будет содержать сообщение об ошибке и ее тип:

```
{
  "Broken": {
    "value": "message error (Parameter 'type error!)"
  }
}
```

5.3. ServerInfoService

Сервис предназначен для получения информации о сервере системы, идентификаторе его сессии работы, временной зоне, а также списке доступных модулей. Описание приведено в файле `server_info.proto`.

5.3.1. Метод `GetModules`

Данный метод предназначен для получения списка модулей сервера системы ПК «Бастион-3» с указанием наименования модуля и его версии. Запрос должен выполняться с использованием токена доступа.

Пример ответа:

```
{
  "modules": [
    {
      "name": "Operators",
      "version": "2024.2"
    },
    {
      "name": "Persons",
      "version": "2024.1"
    },
    ...
  ]
}
```

```
}
```

5.3.2. Метод GetServerState

Запрос на получение информации об идентификаторе сессии работы сервера системы, а также коде его временной зоне. Выполняется с пустым телом запроса. Запрос может быть выполнен без использования токена доступа.

Пример ответа:

```
{
  "Normal": {
    "server_session_uid": "83e711ba-4a1d-4406-a1ed-9b835eacd50f",
    "time_zone_code": 3
  }
}
```

- `server_session_uid` – идентификатор сессии работы сервера системы;
- `time_zone_code` – код временной зоны.

Если произошла ошибка при инициализации запроса, то ответное сообщение выглядит следующим образом:

```
{
  "Broken": {
    "error_message": ""
  }
}
```

5.4. UpdateDataService

Сервис UpdateDataService служит для получения доступа к функционалу изменения данных. Описание сервиса приведено в файле `update_data.proto`.

5.4.1. Метод UpdateData

Запрос на изменение данных системы. Параметры запроса задаются с помощью сообщения UpdateDataRequest. В рамках одного запроса может быть выполнено несколько элементарных изменений, список которых передаётся через поле `operations` сообщения UpdateDataRequest в закодированном в Any виде (подробнее см. в [Приложение 1. Тип google.protobuf.Any](#)). Описание типов сообщений для элементарных изменений будет приведено далее в подразделах. При необходимости задания связей между сущностями, добавляемыми в рамках одного запроса изменения данных, необходимо использовать механизм временных идентификаторов, алгоритм работы которого следующий:

1. На стороне клиента генерируется временное значение для идентификатора добавляемой "родительской" (той, на которую будут ссылаться другие добавляемые сущности) сущности. Временный идентификатор представляет из себя целое знаковое 4-х байтовое число, значение

которого должно быть не больше -100 и должно быть уникальным в рамках всех элементарных операций, передаваемых в одном запросе (если есть другие новые "родительские" сущности, добавляемые в рамках этого же запроса, то для них должны быть сгенерированы другие временные идентификаторы).

2. В элементарной операции добавления "дочерней" сущности в поле, в котором через идентификатор указывается ссылка на "родительскую" сущность (далее поле-ссылка), указывается значение временного идентификатора, сгенерированного на первом шаге.
3. При обработке на сервере системы элементарной операции добавления "родительской" сущности, происходит генерация окончательного идентификатора, при этом строится общая карта соответствия временных и окончательных идентификаторов для всех добавляемых в рамках одного запроса "родительских" сущностей.
4. При обработке операции добавления "дочерней" сущности в случае, если в поле-ссылке задан временный идентификатор (значение не больше -100, все окончательные идентификаторы имеют неотрицательные значения) с помощью карты, заполняемой на 3-м шаге, происходит разрешение временного идентификатора в окончательный: в поле подставляется значение соответствующего окончательного идентификатора.
5. В результатах запроса (сообщение UpdateDataResponse) на клиент возвращается карта соответствия временных и окончательных идентификаторов (поле temp_ids_map). Эта информация может быть использована клиентом для дальнейшей обработки данных.

Запрос изменения данных может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "operations": [
    {
      "type_url": "type.googleapis.com/esprom.taurus.grpc.v1.persons.DeleteAccessLevel",
      "value": "СРЕВ"
    },
    {
      "type_url": "type.googleapis.com/esprom.taurus.grpc.v1.persons.DeleteAccessLevel",
      "value": "СРЕВ"
    }
  ]
}
```

- operations – список элементарных операций изменения данных, которые необходимо выполнить.

Пример ответа на успешное выполнение запроса:

```
{
  "temp_ids_map": {
    "-101": 321,
    "-102": 322
  }
}
```

Если при изменении данных никакие временные идентификаторы не использовались, то поле `temp_ids_map` будет пустым:

```
{
  "temp_ids_map": {}
}
```

В следующих подразделах будет приведено описание сообщений, соответствующих элементарным операциям изменения данных, которые поддерживаются системой.

5.4.1.1. Редактирование уровней доступа

Описание элементарных операций редактирования уровней доступа приведено в файле `persons/access_level_update.proto`. Подробнее про сами уровни доступа можно прочитать в разделе [AccessLevelService](#).

esprom.taurus.grpc.v1.persons.AddAccessLevel

Добавление уровня доступа. Поля:

- `access_level` (тип `esprom.taurus.grpc.v1.persons.AccessLevel`) – описание добавляемого уровня доступа.

esprom.taurus.grpc.v1.persons.DeleteAccessLevel

Удаление уровня доступа. Поля:

- `access_level_id` – идентификатор удаляемого уровня доступа.

esprom.taurus.grpc.v1.persons.UpdateAccessLevel

Обновление уровня доступа. Поля:

- `access_level` (тип `esprom.taurus.grpc.v1.persons.AccessLevel`) – описание обновляемого уровня доступа.

esprom.taurus.grpc.v1.persons.SetAccessLevelContent

Задание состава уровня доступа. Поля:

- `access_level_id` – идентификатор удаляемого уровня доступа;
- `entries` – перечисление элементов состава уровня доступа.

esprom.taurus.grpc.v1.persons.AddWeakAccessLevel

Создание автоматического уровня доступа. Автоматические уровни доступа имеют следующие важные отличия от обычных уровней доступа:

1. Они не могут существовать, если на них нет ссылающихся пропусков. При удалении последнего ссылающегося на пропуски, будет также удалён автоматический уровень доступа.
2. При их добавлении в систему, сначала происходит поиск уже существующего аналогичного по составу автоматического уровня доступа. Если такой уровень будет найден, то добавление не будет фактически выполнено, а переданный временный идентификатор для автоматического уровня доступа будет разрешён в идентификатор уже существующего аналогичного автоматического уровня доступа.

- У автоматического уровня доступа нет явного имени. Оно генерируется автоматически при добавлении сервером системы.

Поля:

- `temp_access_level_id` – временный идентификатор для автоматического уровня доступа, если в системе будет найден аналогичный по составу другой уже существующий автоматический уровень доступа, то его значение будет разрешено в значение идентификатора аналогичного уровня.
- `main_time_block_id` – идентификатор основного временного блока для уровня доступа.
- `entries` – перечисление элементов состава уровня доступа. При добавлении параметров и состава уровня доступа, создается уровень доступа (с параметром `is_weak = true`), если он используется в пропусках или расписаниях.

5.4.1.2. Редактирование карт доступа

Описание элементарных операций редактирования карт доступа приведено в файле `persons/card_update.proto`. Подробнее про сами карты доступа можно прочитать в разделе [CardService](#).

esprom.taurus.grpc.v1.persons.AddCard

Добавление новой карты доступа в систему. Поля:

- `card` (тип `esprom.taurus.grpc.v1.persons.Card`) – описание добавляемой карты доступа.

esprom.taurus.grpc.v1.persons.UpdateCard

Редактирование существующей карты доступа. Поля:

- `card` (тип `esprom.taurus.grpc.v1.persons.Card`) – описание редактируемой карты доступа.

esprom.taurus.grpc.v1.persons.DeleteCard

Удаление существующей карты доступа. Поля:

- `card_id` – идентификатор удаляемой карты доступа.

5.4.1.3. Редактирование словарей и их значений

Описание элементарных операций редактирования словарей и их значений приведено в файле `persons/dictionaries_update.proto`. Подробнее про сами словари и их значения можно прочитать в разделе [DictionariesService](#).

esprom.taurus.grpc.v1.persons.SetDictionaryValue

Операция условного добавления словарного значения. Поля:

- `record` (тип `esprom.taurus.grpc.v1.persons.DictionaryRecord`) – описание словарного значения.

При обработке этой операции сервер системы сначала попытает найти уже существующее аналогичное словарное значение, и, если такое найдётся, добавление не будет выполнено, при этом указанный для значения временный идентификатор будет разрешён в идентификатор найденного аналогичного значения. Например, в системе имеется словарь «Гражданство (`id = 3`)» со значениями "Россия" и "Белоруссия". После выполнения следующих операций (для наглядности в поле `value` используется не закодованное в Ану представление):

```
{
```

```

"operations": [
  {
    "type_url": "type.googleapis.com/esprom.taurus.grpc.v1.persons.SetDictionaryValue",
    "value": {
      {
        "record": {
          "header_id": 3,
          "id": -101,
          "is_system": false,
          "value": "Белоруссия"
        }
      }
    },
    {
      "type_url": "type.googleapis.com/esprom.taurus.grpc.v1.persons.SetDictionaryValue",
      "value": {
        "record": {
          "header_id": 3,
          "id": -102,
          "is_system": false,
          "value": "Казахстан"
        }
      }
    }
  ]
}

```

Будет добавлено только значение «Казахстан», так как такого значения ещё нет в системе.

esprom.taurus.grpc.v1.persons.AddDictionaryRecord

Операция добавления словарного значения. Если в системе уже есть такое значение, то операция завершится с ошибкой. Поля:

- record (тип `esprom.taurus.grpc.v1.persons.DictionaryRecord`) – описание словарного значения.

esprom.taurus.grpc.v1.persons.UpdateDictionaryRecord

Операция редактирования словарного значения. Если в системе уже есть значение, совпадающее с новым значением, то операция завершится с ошибкой. Поля:

- record (тип `esprom.taurus.grpc.v1.persons.DictionaryRecord`) – описание словарного значения.

esprom.taurus.grpc.v1.persons.DeleteDictionaryRecord

Операция удаления словарного значения. Поля:

- record_id – идентификатор удаляемого словарного значения.

esprom.taurus.grpc.v1.persons.AddDictionaryHeader

Операция добавления словаря. Поля:

- header (тип `esprom.taurus.grpc.v1.persons.DictionaryHeader`) – описание словаря.

esprom.taurus.grpc.v1.persons.UpdateDictionaryHeader

Операция редактирования словаря. Поля:

- header (тип `esprom.taurus.grpc.v1.persons.DictionaryHeader`) – описание словаря.

esprom.taurus.grpc.v1.persons.DeleteDictionaryHeader

Операция удаления словаря. Поля:

- `header_id` – идентификатор удаляемого словаря.

5.4.1.4. Редактирование пропусков

Описание элементарных операций редактирования пропусков приведено в файле `persons/pass_update.proto`. Подробнее про пропуска можно прочитать в разделе [PassService](#).

`esprom.taurus.grpc.v1.persons.AddPass`

Операция добавления пропуска. Поля:

- `pass` (тип `esprom.taurus.grpc.v1.persons.Pass`) – описание добавляемого пропуска.

`esprom.taurus.grpc.v1.persons.UpdatePass`

Операция редактирования пропуска. Поля:

- `pass` (тип `esprom.taurus.grpc.v1.persons.Pass`) – описание редактируемого пропуска.

`esprom.taurus.grpc.v1.persons.DeletePass`

Операция удаления пропуска. Поля:

- `pass_id` – идентификатор удаляемого пропуска.

`esprom.taurus.grpc.v1.persons.UpdatePassSecurityControlGroupSetting`

Операция задания для пропуска группы управления охраной. Поля:

- `group_settings` (тип `esprom.taurus.grpc.v1.persons.PassSecurityControlGroupSetting`) – настройка привязки пропуска к группе управления охраной.

`esprom.taurus.grpc.v1.persons.DeletePassSecurityControlGroupSetting`

Операция сброса для пропуска группы управления охраны. Поля:

- `pass_id` – идентификатор пропуска.

5.4.1.5. Редактирование персон

Описание элементарных операций редактирования персон приведено в файле `persons/person_update.proto`. Подробнее про персон можно прочитать в разделе [PersonService](#).

`esprom.taurus.grpc.v1.persons.AddPerson`

Операция добавления персоны. Поля:

- `person` (тип `esprom.taurus.grpc.v1.persons.Person`) – описание добавляемой персоны.

`esprom.taurus.grpc.v1.persons.UpdatePerson`

Операция редактирования персоны. Поля:

- `person` (тип `esprom.taurus.grpc.v1.persons.Person`) – описание редактируемой персоны.

`esprom.taurus.grpc.v1.persons.DeletePerson`

Операция удаления персоны. Поля:

- `person_id` – идентификатор удаляемой персоны.

`esprom.taurus.grpc.v1.persons.UpdatePersonallInfo`

Операция обновления персональных данных персоны. Поля:

- `personal_info` (тип `esprom.taurus.grpc.v1.persons.PersonalInfo`) – персональные данные/реквизиты персоны.

`esprom.taurus.grpc.v1.persons.SetPersonPhoto`

Операция задания фото для персоны. Поля:

- `person_id` – идентификатор персоны;
- `photo_data` – фото персоны.

`esprom.taurus.grpc.v1.persons.DeletePersonPhoto`

Операция сброса фото для персоны. Поля:

- `person_id` – идентификатор персоны.

`esprom.taurus.grpc.v1.persons.UpdatePersonAdditionalField`

Операция обновления настроек дополнительного поля персоны. Поля:

- `additional_field` (тип `esprom.taurus.grpc.v1.persons.PersonAdditionalField`) – настройки дополнительного поля персоны.

5.4.1.6. Редактирование временных блоков

Описание элементарных операций редактирования временных блоков приведено в файле `persons/time_block_update.proto`. Подробнее про временные блоки можно прочитать в разделе [TimeBlockService](#).

`esprom.taurus.grpc.v1.persons.AddTimeBlock`

Операция добавления временного блока. Поля:

- `time_block` (тип `esprom.taurus.grpc.v1.persons.TimeBlock`) – описание добавляемого временного блока.

`esprom.taurus.grpc.v1.persons.UpdateTimeBlock`

Операция редактирования временного блока. Поля:

- `time_block` (тип `esprom.taurus.grpc.v1.persons.TimeBlock`) – описание редактируемого временного блока.

`esprom.taurus.grpc.v1.persons.DeleteTimeBlock`

Операция редактирования временного блока. Поля:

- `time_block_id` – идентификатор удаляемого временного блока.

`esprom.taurus.grpc.v1.persons.AddWeakTimeBlock`

Создание автоматического временного блока. Автоматические временные блоки имеют следующие важные отличия от обычных временных блоков:

1. Они не могут существовать, если на них нет ссылающихся сущностей (уровни доступа и/или их элементы, территории). При удалении последней ссылки на автоматический временной блок, он будет удалён.
2. При их добавлении в систему, сначала происходит поиск уже существующего аналогичного автоматического временного блока. Если такой блок будет найден, то добавление не будет

фактически выполнено, а переданный временный идентификатор для автоматического временного блока будет разрешён в идентификатор уже существующего аналогичного автоматического временного блока.

3. У автоматического временного блока нет явного имени. Оно автоматически генерируется сервером системы при добавлении.
4. Автоматические временные блоки всегда являются недельными и учитывают праздники.

Поля:

- `temp_time_block_id` – временный идентификатор для автоматического временного блока, если в системе будет найден аналогичный по составу другой уже существующий автоматический временной блок, то его значение будет разрешено в значение идентификатора аналогичного блока.
- `time_zones` – перечисление временных зон входящих во временной блок.

5.4.1.7. Редактирование дополнительных полей устройств

Описание элементарных операций редактирования значений дополнительных полей устройств приведено в файле `drivers/additional_field_update.proto`. Подробнее про временные блоки можно прочесть в разделе [AdditionalFieldService](#).

esprom.taurus.grpc.v1.persons.UpdateAdditionalFieldValue

Операция задания временного блока. Поля:

- `value` (тип `esprom.taurus.grpc.v1.drivers.AdditionalFieldValue`) – описание задаваемого значения дополнительного поля устройства.

5.4.1.8. Редактирование организационной структуры

Описание элементарных операций редактирования организационной структуры приведено в файле `persons/organization_structure_update.proto`. Подробнее про организационную структуру можно прочитать в разделе [OrganizationStructureService](#).

esprom.taurus.grpc.v1.persons.AddOrganizationNode

Операция добавления узла дерева организационной структуры. Поля:

- `node` (тип `esprom.taurus.grpc.v1.persons.OrganizationNode`) – описание добавляемого узла дерева организационной структуры.

esprom.taurus.grpc.v1.persons.UpdateOrganizationNode

Операция редактирования узла дерева организационной структуры. Поля:

- `node` (тип `esprom.taurus.grpc.v1.persons.OrganizationNode`) – описание редактируемого узла дерева организационной структуры.

esprom.taurus.grpc.v1.persons.DeleteOrganizationNode

Операция удаления узла дерева организационной структуры. Поля:

- `node_id` – идентификатор удаляемого узла дерева организационной структуры.

esprom.taurus.grpc.v1.persons.SetOrganizationNode

Операция условного добавления узла дерева организационной структуры. Поля:

- node (тип `esprom.taurus.grpc.v1.persons.OrganizationNode`) – описание добавляемого узла дерева организационной структуры.

При обработке этой операции сервер системы сначала попытает найти уже существующий аналогичный узел дерева организационной структуры, и, если такой найдётся, добавление не будет выполнено, при этом указанный для узла временный идентификатор будет разрешён в идентификатор найденного аналогичного узла. При поиске аналогичного узла выполняется сравнение по трём полям: родительскому узлу, типу и имени узла.

Например, если текущее дерево организационной структуры выглядит следующим образом:

- Все (id: 0)
 - ООО Организация 1 (id: 101)
 - Департамент 1 (id: 103)
 - Департамент 2 (id: 104)
 - ООО Организация 2 (id: 102)

```
{
  "operation": [
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.persons.SetOrganizationNode",
      "value": {
        "node": {
          "id": -101,
          "name": "ООО Организация 2",
          "node_type": "ORGANIZATION_NODE_TYPE_ORGANIZATION",
          "parent_id": {
            "value": 0
          }
        }
      }
    },
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.persons.SetOrganizationNode",
      "value": {
        "node": {
          "id": -102,
          "name": "ООО Департамент 1",
          "node_type": "ORGANIZATION_NODE_TYPE_DEPARTMENT",
          "parent_id": {
            "value": -101
          }
        }
      }
    }
  ]
}
```

Систему будет добавлен только «Департамент 2», который является дочерним в организации «ООО Организация 2». Организация не будет добавлена, так как она уже есть в системе.

`esprom.taurus.grpc.v1.persons.AllowPassCategoryForOrganizationNode`

Операция разрешения использования категории пропусков для узла дерева организационной структуры. Поля:

- `organization_node_id` – идентификатор узла организационной структуры, для которого разрешают использование категории пропусков.
- `pass_category_id` – идентификатор категории пропусков, которая должна стать доступной для использования в узле организационной структуры. Если значение не задано, то для узла организационной структуры станут доступны все категории пропусков.

`esprom.taurus.grpc.v1.persons.RestrictPassCategoryForOrganizationNode`

Операция изменения данных, которая запрещает использование категории пропусков для узла организационной структуры. Поля:

- `organization_node_id` – идентификатор узла организационной структуры, для которого запрещают использование категории пропусков.
- `pass_category_id` – идентификатор категории пропусков, которую запрещают для использования в узле организационной структуры. Если значение не задано, то для узла организационной структуры будут запрещены все категории пропусков.

5.4.1.9. Редактирование транспортных средств и их статусов

Описание элементарных операций редактирования статусов транспортных средств / транспортных средств приведено в файле `mtp/transport_update.proto`. Подробнее про статусы транспортных средств и транспортные средства можно прочитать в разделе [TransportService](#).

`esprom.taurus.grpc.v1.mtp.AddTransportStatus`

Операция добавления статуса транспортного средства. Поля:

- `status` (тип `esprom.taurus.grpc.v1.mtp.TransportStatus`) – описание добавляемого статуса транспортного средства.

`esprom.taurus.grpc.v1.mtp.UpdateTransportStatus`

Операция редактирования статуса транспортного средства. Поля:

- `status` (тип `esprom.taurus.grpc.v1.mtp.TransportStatus`) – описание редактируемого статуса транспортного средства.

`esprom.taurus.grpc.v1.mtp.DeleteTransportStatus`

Операция удаления статуса транспортного средства. Поля:

- `id` – идентификатор статуса транспортного средства.

`esprom.taurus.grpc.v1.mtp.AddTransport`

Операция добавления транспортного средства. Поля:

- `transport` (тип `esprom.taurus.grpc.v1.mtp.Transport`) – описание добавляемого транспортного средства.

`esprom.taurus.grpc.v1.mtp.UpdateTransport`

Операция редактирования транспортного средства. Поля:

- `transport` (тип `esprom.taurus.grpc.v1.mtp.Transport`) – описание редактируемого транспортного средства.

esprom.taurus.grpc.v1.mtp.DeleteTransport

Операция удаления транспортного средства. Поля:

- `id` – идентификатор транспортного средства.

esprom.taurus.grpc.v1.mtp.TransportPhotoUpdateOperation

Операция задания фото для транспортного средства. Поля:

- `transport_id` – идентификатор транспортного средства;
- `photo_data` – фото транспортного средства.

esprom.taurus.grpc.v1.mtp.TransportPhotoDeleteOperation

Операция сброса фото для транспортного средства. Поля:

- `transport_id` – идентификатор транспортного средства.

5.4.1.10. Редактирование операторов, ролей операторов и полномочий

Описание элементарных операций редактирования приведено в файле `operators/operators_info_update`. Подробнее про роли операторов и их полномочия можно прочитать в разделе [OperatorsInfoService](#).

esprom.taurus.grpc.v1.operators.AddRole

Операция добавления роли оператора. Поля:

- `role` (тип `esprom.taurus.grpc.v1.operators.Role`) – описание добавляемой роли оператора.

esprom.taurus.grpc.v1.operators.UpdateRole

Операция редактирования роли операторов. Поля:

- `role` (тип `esprom.taurus.grpc.v1.operators.Role`) – описание редактируемой роли оператора.

esprom.taurus.grpc.v1.operators.DeleteRole

Операция удаления роли операторов. Поля:

- `role_id` (тип `int32`) – идентификатор удаляемой роли операторов.

esprom.taurus.grpc.v1.operators.AddOperator

Операция добавления оператора. Поля:

- `operator` (тип `esprom.taurus.grpc.v1.operators.Operator`) – описание добавляемого оператора.

esprom.taurus.grpc.v1.operators.UpdateOperator

Операция редактирования оператора. Поля:

- `operator` (тип `esprom.taurus.grpc.v1.operators.Operator`) – описание редактируемого оператора.

esprom.taurus.grpc.v1.operators.DeleteOperator

Операция удаления оператора. Поля:

- `operator_id` (тип `int32`) – идентификатор удаляемого оператора.

esprom.taurus.grpc.v1.operators.UpdateOperatorInfo

Операция редактирования дополнительных атрибутов оператора. Поля:

- `operator_info` (тип `esprom.taurus.grpc.v1.operators.OperatorInfo`) – описание дополнительных атрибутов оператора.

`esprom.taurus.grpc.v1.operators.UpdateOperatorPassword`

Операция установки/обновления пароля оператора. Поля:

- `operator_id` (тип `int32`) – описание оператора;
- `new_password` (тип `string`) - новый пароль.

`esprom.taurus.grpc.v1.operators.UpdateGeneralDevicePermissions`

Операция обновления общесистемных наследуемых полномочий роли оператора для устройства. Поля:

- `general_device_permissions` (тип `esprom.taurus.grpc.v1.operators.GeneralDevicePermissions`) – общесистемных наследуемых полномочий для устройства.

`esprom.taurus.grpc.v1.operators.DeleteGeneralDevicePermissions`

Операция удаления общесистемных наследуемых полномочий роли оператора для устройства. После удаления, устройство начнёт наследовать полномочия от родительского устройства. Поля:

- `role_id` (тип `int32`) – идентификатор роли;
- `sdn` (тип `int32`) – идентификатор устройства.

`esprom.taurus.grpc.v1.operators.UpdateRestrictedDeviceActions`

Операция обновления запрещённых для роли действий для устройства. Поля:

- `restricted_device_actions` (тип `esprom.taurus.grpc.v1.operators.RestrictedDeviceActions`) – описание запрещённых для роли оператора действий для устройства.

`esprom.taurus.grpc.v1.operators.UpdateRoleBastionSettings`

Операция обновления настроек Поста Охраны для роли оператора. Поля:

- `role_bastion_settings` (тип `esprom.taurus.grpc.v1.operators.RoleBastionSettings`) – описание настроек Поста Охраны для роли оператора.

`esprom.taurus.grpc.v1.operators.AllowApplicationPermission`

Операция добавления полномочия приложений для роли. Поля:

- `permission` (тип `esprom.taurus.grpc.v1.operators.ApplicationPermission`) – описание добавляемого для роли оператора полномочия.

`esprom.taurus.grpc.v1.operators.RestrictApplicationPermission`

Операция удаления полномочия приложений для роли. Поля:

- `permission` (тип `esprom.taurus.grpc.v1.operators.ApplicationPermission`) – описание удаляемого для роли оператора полномочия.

5.4.1.11. Редактирование рабочих дней

Описание элементарных операций редактирования приведено в файле `attendance/regime_update.proto`. Подробнее про рабочие дни можно прочитать в разделе [RegimeService](#).

`esprom.taurus.grpc.v1.attendance.AddRegime`

Операция добавления рабочего дня. Поля:

- regime (тип `esprom.taurus.grpc.v1.attendance.Regime`) – описание добавляемого рабочего дня.

esprom.taurus.grpc.v1.attendance.UpdateRegime

Операция редактирования рабочего дня. Поля:

- regime (тип `esprom.taurus.grpc.v1.attendance.Regime`) – описание редактируемого рабочего дня.

esprom.taurus.grpc.v1.attendance.DeleteRegime

Операция удаления рабочего дня. Поля:

- regime_id (тип `int32`) – идентификатор удаляемого рабочего дня.

5.4.1.12. Редактирование графиков сменности и позиций графиков сменности

Описание элементарных операций редактирования приведено в файле `attendance/shift_schedule_update.proto`. Подробнее про графики сменности и позиции графиков сменности можно прочитать в разделе [ShiftScheduleService](#).

esprom.taurus.grpc.v1.attendance.AddShiftSchedule

Операция добавления графика сменности. Поля:

- shift_schedule (тип `esprom.taurus.grpc.v1.attendance.ShiftSchedule`) – описание добавляемого графика сменности.

esprom.taurus.grpc.v1.attendance.UpdateShiftSchedule

Операция редактирования графика сменности. Поля:

- shift_schedule (тип `esprom.taurus.grpc.v1.attendance.ShiftSchedule`) – описание редактируемого графика сменности.

esprom.taurus.grpc.v1.attendance.DeleteShiftSchedule

Операция удаления графика сменности. Поля:

- id (тип `int32`) – идентификатор удаляемого графика сменности.

esprom.taurus.grpc.v1.attendance.AddShiftScheduleItem

Операция добавления позиции графика сменности. Поля:

- shift_schedule_item (тип `esprom.taurus.grpc.v1.attendance.ShiftScheduleItem`) – описание добавляемой позиции графика сменности.

esprom.taurus.grpc.v1.attendance.UpdateShiftScheduleItem

Операция редактирования позиции графика сменности. Поля:

- shift_schedule_item (тип `esprom.taurus.grpc.v1.attendance.ShiftScheduleItem`) – описание редактируемой позиции графика сменности.

esprom.taurus.grpc.v1.attendance.DeleteShiftScheduleItem

Операция удаления позиции графика сменности. Поля:

- id (тип `int32`) – идентификатор удаляемой позиции графика сменности.

esprom.taurus.grpc.v1.attendance.AllowShiftScheduleOrganization

Операция добавления настройки признака значения по умолчанию графика сменности для организации/подразделения. Поля:

- `shift_schedule_organization` (тип `esprom.taurus.grpc.v1.attendance.ShiftScheduleOrganization`) – описание добавляемого для графика сменности признака значения по умолчанию для организации/подразделения.

`esprom.taurus.grpc.v1.attendance.RestrictShiftScheduleOrganization`

Операция удаления настройки признака значения по умолчанию графика сменности для организации/подразделения. Поля:

- `shift_schedule_organization` (тип `esprom.taurus.grpc.v1.attendance.ShiftScheduleOrganization`) – описание добавляемого для графика сменности признака значения по умолчанию для организации/подразделения.



Внимание!

Для создания графика сменности необходимо в рамках одной операции добавлять и сам график и хотя бы одну его позицию.

5.4.1.13. Редактирование трудовых договоров

Описание элементарных операций редактирования приведено в файле `attendance/work_contract_update.proto`. Подробнее про трудовые договора можно прочитать в разделе [WorkContractService](#).

`esprom.taurus.grpc.v1.attendance.AddWorkContract`

Операция добавления трудового договора. Поля:

- `work_contract` (тип `esprom.taurus.grpc.v1.attendance.WorkContract`) – описание добавляемого трудового договора.

`esprom.taurus.grpc.v1.attendance.UpdateWorkContract`

Операция редактирования трудового договора. Поля:

- `work_contract` (тип `esprom.taurus.grpc.v1.attendance.WorkContract`) – описание редактируемого трудового договора.

`esprom.taurus.grpc.v1.attendance.DeleteWorkContract`

Операция удаления трудового договора. Поля:

- `id` (тип `int32`) – идентификатор удаляемого трудового договора.

`esprom.taurus.grpc.v1.attendance.UpdatePassCategoryAtdSettings`

Операция редактирования настройки параметров учета рабочего времени для категории пропуска. Поля:

- `pass_category_atd_settings` (тип `esprom.taurus.grpc.v1.attendance.PassCategoryAtdSettings`) – описание изменяемого для категории пропуска параметра учета рабочего времени.

5.4.1.14. Редактирование дополнительных полей пропусков

Описание элементарных операций редактирования дополнительных полей устройств приведено в файле `persons/pass_additional_field_update.proto`. Подробнее про дополнительные поля пропуска можно прочесть в разделе [PassAdditionalFieldService](#).

esprom.taurus.grpc.v1.persons.UpdatePassAdditionalFieldValue

Операция обновления значения дополнительного поля. Поля:

- `value` (тип `esprom.taurus.grpc.v1.persons.PassAdditionalFieldValue`) – описание обновляемого значения дополнительного поля пропуска.

esprom.taurus.grpc.v1.persons.DeletePassAdditionalFieldValue

Операция удаления значения дополнительного поля. Поля:

- `pass_id` – идентификатор пропуска;
- `descriptor_id` – идентификатор дескриптора дополнительного поля пропуска.

esprom.taurus.grpc.v1.persons.AddPassAdditionalFieldDescriptor

Операция добавления дескриптора дополнительного поля. Поля:

- `field_descriptor` (тип `esprom.taurus.grpc.v1.persons.PassAdditionalFieldDescriptor`) – описание добавляемого дескриптора дополнительного поля пропуска.

esprom.taurus.grpc.v1.persons.UpdatePassAdditionalFieldDescriptor

Операция обновления дескриптора дополнительного поля. Поля:

- `field_descriptor` (тип `esprom.taurus.grpc.v1.persons.PassAdditionalFieldDescriptor`) – описание обновляемого дескриптора дополнительного поля пропуска.

esprom.taurus.grpc.v1.persons.DeletePassAdditionalFieldDescriptor

Операция удаления дескриптора дополнительного поля. Поля:

- `descriptor_id` – идентификатор дескриптора дополнительного поля пропуска.

5.4.1.15. Редактирование привязки устройств к территориям

Описание элементарных операций редактирования привязки устройств к территориям приведено в файле `persons/device_control_area_link_update.proto`. Подробнее про привязку устройств к территориям можно прочесть в разделе [DeviceControlAreaLinkService](#).

esprom.taurus.grpc.v1.persons.UpdateDeviceControlAreaLink

Операция задания явной привязки устройства к территории.

Поля:

- `device_control_area_link` (тип `esprom.taurus.grpc.v1.persons.DeviceControlAreaLink`) – описание привязки устройства к территории.

esprom.taurus.grpc.v1.persons.DeleteDeviceControlAreaLink

Операция удаления явной привязки устройства к территории (включение наследования значения привязки от родительского устройства). Удаление привязки доступно только для не корневых устройств в иерархии.

Поля:

- sdn (тип int32) – идентификатор устройства.

5.4.1.16. Редактирование биометрических шаблонов

Описание элементарных операций редактирования биометрических шаблонов приведено в файле `persons/biometry_update.proto`. Подробнее про биометрические шаблоны можно прочитать в разделе [BiometryService](#).

esprom.taurus.grpc.v1.persons.AddBioTemplate

Операция добавления биометрического шаблона. Поля:

- bio_template (тип `esprom.taurus.grpc.v1.persons.BioTemplate`) – описание добавляемого биометрического шаблона.

5.4.1.17. Редактирование настроек филиала репликации и настроек репликации пропусков

Описание элементарных операций редактирования настроек филиала репликации и настроек репликации пропусков приведено в файле `replication/replication_branch_update.proto`. Подробнее про настройки филиала репликации и настройки репликации пропусков можно прочитать в разделе [ReplicationBranchService](#).

esprom.taurus.grpc.v1.replication.AddBranch

Операция добавления филиала репликации. Поля:

- branch_info (тип `esprom.taurus.grpc.v1.replication.BranchInfo`) – описание добавляемого филиала репликации.

esprom.taurus.grpc.v1.replication.UpdateBranch

Операция редактирования филиала репликации. Поля:

- branch_info (тип `esprom.taurus.grpc.v1.replication.BranchInfo`) – описание редактируемого филиала репликации.

esprom.taurus.grpc.v1.replication.DeleteBranch

Операция удаления филиала репликации. Поля:

- branch_id – идентификатор удаляемого филиала репликации;
- pickup_branch_data – признак, указывающий, что локальный филиал необходимо назначить владельцем данных удаляемого филиала.

esprom.taurus.grpc.v1.replication.AddGlobalAccessLevel

Операция добавления глобального уровня доступа. Поля:

- global_access_level_info (тип `esprom.taurus.grpc.v1.replication.GlobalAccessLevelInfo`) – описание добавляемого глобального уровня доступа.

esprom.taurus.grpc.v1.replication.UpdateGlobalAccessLevel

Операция редактирования глобального уровня доступа. Поля:

- global_access_level_info (тип `esprom.taurus.grpc.v1.replication.GlobalAccessLevelInfo`) – описание редактируемого глобального уровня доступа.

esprom.taurus.grpc.v1.replication.DeleteGlobalAccessLevel

Операция удаления глобального уровня доступа. Поля:

- `global_access_level_id` – идентификатор удаляемого глобального уровня доступа.

esprom.taurus.grpc.v1.replication.UpdatePassReplicationProperties

Операция редактирования настроек репликации персонального пропуска. Данная операция позволяет задавать для пропуска глобальный уровень доступа, список филиалов – пунктов назначения, а также отдельные назначения глобальных уровней доступа для филиалов – пунктов назначения. Поля:

- `replication_properties` (тип `esprom.taurus.grpc.v1.replication.PassReplicationPropertiesInfo`) – настройки репликации пропуска.

esprom.taurus.grpc.v1.replication.UpdatePassCategoryOrganizationReplicationProperties

Операция редактирования настроек репликации по умолчанию для персональных пропусков заданной категории и организации/подразделения. Поля:

- `replication_properties` (тип `esprom.taurus.grpc.v1.replication.PassCategoryOrganizationReplicationProperties`) – настройки репликации по умолчанию для персональных пропусков указанной категории и организации/подразделения.

esprom.taurus.grpc.v1.replication.UpdateDefaultConflictResolveRule

Операция редактирования существующего правила разрешения конфликта по умолчанию. Поля:

- `conflict_resolve_rule` (тип `esprom.taurus.grpc.v1.replication.ConflictResolveRule`) – правило разрешения конфликта.

esprom.taurus.grpc.v1.replication.UpdateForeignDataEditSettings

Операция редактирования параметров редактирования "чужих" данных, полученных филиалом через репликацию. Поля:

- `settings` (тип `esprom.taurus.grpc.v1.replication.ForeignDataEditSettings`) – параметры редактирования "чужих" данных.

5.5. AccessLevelService

Файл `access_level.proto` предназначен для получения информации об уровнях доступа.

5.5.1. Метод `GetAccessLevels`

Запрос на получение списка уровней доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "access_levels": [
    {
      "id": 141,
      "physical_number": {
        "value": 4
      },
      "is_weak": true,
    }
  ]
}
```

```
    "name": "Уровень доступа №1 (авто)",
    "main_time_block_id": 4
  },
  {
    "id": 1,
    "physical_number": null,
    "is_weak": false,
    "name": "По умолчанию",
    "main_time_block_id": 4
  },
  {
    "id": 121,
    "physical_number": null,
    "is_weak": false,
    "name": "Уровень доступа 3",
    "main_time_block_id": 1
  },
  ...
]
}
```

Ответное сообщение содержит список всех уровней доступа в системе. О каждом уровне доступа представлена следующая информация:

- `id` – уникальный идентификатор;
- `physical_number` – физический номер. Если уровень доступа не используется значение будет `null`;
 - `value` – значение физического номера;
- `is_weak` – булево значение указывающий на то что уровень доступа задан был сгенерирован автоматически в ходе ручного задания состава уровня доступа;
- `name` – наименование уровня доступа, максимальная длина 100 символов;
- `main_time_block_id` – идентификатор временного блока используемого в данном уровне доступа.

5.5.2. Метод `GetAccessLevel`

Запрос на получение информации об уровне доступа. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "access_level_id": 101
}
```

В тело запроса необходимо записать в поле `access_level_id` уникальный идентификатор уровня доступа, информацию о котором необходимо получить.

Пример ответа:

```
{
  "access_level": {
    "id": 101,

```

```

    "physical_number": {
      "value": 4
    },
    "is_weak": true,
    "name": "Уровень доступа №1 (авто)",
    "main_time_block_id": 4
  }
}

```

В теле *access_level* указаны та же информация об уровне доступа, что и в ответном сообщении на запрос *GetAccessLevels*.

5.5.3. Метод *GetAccessLevelContents*

Запрос на получение информации о составе уровня доступа. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "access_level_id": 101
}

```

В тело запроса необходимо записать в поле *access_level_id* уникальный идентификатор уровня доступа, информацию о составе которого необходимо получить. Если в состав уровня доступа пуст, то ответные сообщения не будут получены, при этом запрос выполнится без ошибок.

Ответ разделен на несколько сообщений. Сообщения разделены на несколько категорий:

Сообщение о вложенном считывателе:

```

{
  "entry": {
    "Reader": {
      "reader_sdn": 350,
      "custom_time_block_id": {
        "value": 8
      }
    }
  }
}

```

В сообщении содержится:

- *reader_sdn* – уникальный идентификатор считывателя;
- *custom_time_block_id* – уникальный идентификатор используемого временного блока. Если к считывателю привязан автоматический временной блок, то значение будет равно *null*.

Сообщение о вложенной территории:

```

{

```

```

    "entry": {
      "ControlArea": {
        "control_area_id": 100,
        "custom_time_block_id": {
          "value": 5
        }
      }
    }
  }
}

```

- reader_sdn – уникальный идентификатор территории;
- custom_time_block_id – уникальный идентификатор используемого временного блока. Если к считывателю привязан автоматический временной блок, то значение будет равно *null*.

Сообщение о вложенном уровне доступа:

```

{
  "entry": {
    "NestedLevel": {
      "access_level_id": 126,
      "order": 4
    }
  }
}

```

- access_level_id – уникальный идентификатор вложенного уровня доступа;
- order – порядковый номер.

5.5.4. Метод `GetAccessLevelContentCompositions`

Запрос на получение информации о считывателях входящих в состав уровня доступа. Запрос может быть выполнен только с использованием токена доступа. Если в состав уровня доступа пуст, то ответные сообщения не будут получены, при этом запрос выполнится без ошибок.

Пример запроса:

```

{
  "access_level_id": 101
}

```

В тело запроса необходимо записать в поле `access_level_id` уникальный идентификатор уровня доступа, информацию о вложенных считывателях которых необходимо получить.

В качестве ответа будут получены несколько сообщений, в зависимости от количества вложенных считывателя.

Пример ответа:

```

{
  "reader_sdn": 350,
  "time_block_id": 8
}

```

```
}
```

- reader_sdn – уникальный идентификатор считывателя;
- custom_time_block_id – уникальный идентификатор используемого временного блока.

5.5.5. Метод AccessLevelChanged

Запрос на отслеживание изменений в уровнях доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример сообщений:

```
{  
  "access_level_id": 122,  
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"  
}
```

- access_level_id – уникальный идентификатор уровня доступа, которого затронули изменения;
- change_type – тип произведенных изменений. Это может быть добавление (ENTITY_CHANGE_TYPE_ADD), обновление (ENTITY_CHANGE_TYPE_UPDATE), удаление (ENTITY_CHANGE_TYPE_DELETE) или неизвестное изменение (ENTITY_CHANGE_TYPE_UNSPECIFIED).

5.5.6. Метод AccessLevelContentChanged

Запрос на отслеживание изменений в составах уровней доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример сообщения:

```
{  
  "access_level_id": 122,  
}
```

- access_level_id – уникальный идентификатор уровня доступа, состав которого был изменен.

5.6. CardService

Файл card.proto предназначен для получения информации о картах доступа.

5.6.1. Метод GetCard

Запрос на получение информации о карте доступа. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса с использованием идентификатора доступа:

```
{
  "by_card_id": {
    "card_id": 313
  },
}
```

Если карта не была найдена, то ответ будет следующим:

```
{
  "card": null
}
```

Если карта с таким номером существует, то ответ содержит информацию о карте:

```
{
  "card": {
    "id": 313,
    "full_card_code": "1104801",
    "serial_number": {
      "value": "345"
    },
    "create_date": {
      "seconds": "1712853501",
      "nanos": 340667000
    },
    "return_date": {
      "seconds": "1712903024",
      "nanos": 121046000
    },
    "status": "CARD_STATUS_ISSUED",
    "identifier_type": 0,
    "mifare_security_level": 0,
    "pre_issued": false,
    "comments": null
  }
}
```

- `id` – уникальный идентификатор карты доступа;
- `full_card_code` – полный номер карты;
- `serial_number` – содержит числовое значение серийного номера карты;
- `create_date` – дата создания карты доступа, значение хранит время в секундах *seconds* и наносекундах *nanos*;
- `return_date` – дата возврата карты доступа, значение хранит время в секундах *seconds* и наносекундах *nanos*;
- `status` – указывает текущий статус пропуска (выпущен `CARD_STATUS_ISSUED`, потерян `CARD_STATUS_LOST` и другие, подробнее о статусах написано в документации «Бастион-3 – Бюро пропусков. Руководство оператора»;

- `identifier_type` – тип идентификатора карты, указывающий об используемом считывателе. Подробнее о типах идентификации написано в документации «Бастион-3 – Бюро пропусков. Руководство оператора»;
- `mifare_security_level` – номер уровня безопасности для карт доступа MIFARE;
- `pre_issued` – показывает, была ли карта эмитирована в ПК «Бастион-3»;
- `comments` - комментарий

Помимо поиска по идентификатору карты, можно отправить запрос на получение информации о карте с использованием серийного номера карты:

```
{
  "by_serial_number": {
    "serial_number": "345"
  }
}
```

Ответное сообщение аналогично запросу карты доступа по идентификатору карты.

5.6.2. Метод GetCards

Запрос на получение списка карт доступа. Запрос может быть выполнен только с использованием токена доступа.

Для получения списка карты есть несколько запросов.

Запрос с использованием кода карт:

```
{
  "by_card_code": {
    "card_code": "4"
  }
}
```

В запросе требуется указать код карты доступа. Ответ содержит список всех карт, у которых указан искомый код карты:

```
{
  "cards": [
    {
      "id": 413,
      "full_card_code": "4",
      "serial_number": null,
      "create_date": {
        "seconds": "1712906246",
        "nanos": 787102000
      },
      "return_date": {
        "seconds": "1712906783",
        "nanos": 432439000
      },
      "status": "CARD_STATUS_UNUSABLE",
    }
  ]
}
```

```

        "identifier_type": 0,
        "mifare_security_level": 0,
        "pre_issued": false,
        "comments": null
    },
    {
        "id": 433,
        "full_card_code": "4",
        "serial_number": null,
        "create_date": {
            "seconds": "1712906808",
            "nanos": 882140000
        },
        "return_date": null,
        "status": "CARD_STATUS_ISSUED",
        "identifier_type": 0,
        "mifare_security_level": 0,
        "pre_issued": false,
        "comments": null
    },
    ...
]
}

```

Для того, чтобы получить список конкретных карт доступа, можно использовать запрос по идентификаторам карт доступа:

```

{
  "by_card_ids": {
    "card_ids": [
      313,
      433
    ]
  }
}

```

В тело *card_ids* требуется указать список идентификаторов карт доступа, информацию о которых требуется получить.

Если карт доступа с указанными идентификаторами нет, то ответ будет содержать пустой список:

```

{
  "cards": []
}

```

При наличии искомых карт, ответное сообщение содержит список карт доступа:

```

{
  "cards": [
    {
      "id": 313,
      "full_card_code": "55328922",
      "serial_number": null,

```

```

        "create_date": {
            "seconds": "1712906246",
            "nanos": 787102000
        },
        "return_date": {
            "seconds": "1712906783",
            "nanos": 432439000
        },
        "status": "CARD_STATUS_UNUSABLE",
        "identifier_type": 0,
        "mifare_security_level": 0,
        "pre_issued": false,
        "comments": null
    },
    {
        "id": 433,
        "full_card_code": "438175",
        "serial_number": null,
        "create_date": {
            "seconds": "1712906808",
            "nanos": 882140000
        },
        "return_date": null,
        "status": "CARD_STATUS_ISSUED",
        "identifier_type": 0,
        "mifare_security_level": 0,
        "pre_issued": false,
        "comments": null
    },
    ...
]
}

```

Для получения списка всех карт требуется отправить запрос:

```

{
  "empty": {}
}

```

Для получения списка свободных карт нужно использовать запрос:

```

{
  "free_cards": {
    "packet_size": 3
  }
}

```

Ответ посылается несколькими пакетами информации, каждый пакет содержит количество карт, указанные в поле *packet_size*. В качестве завершающего пакета данных, будет получен пустой список карт:

```

{
  "cards": []
}

```

```
}
```

5.6.3. Метод ReturnCardToNormalState

Запрос на возврат в обращения ранее изъятой из оборота карты доступа. Запрос может быть выполнен только с использованием токена доступа. В запросе требуется указать идентификатор ранее изъятой из оборота карты доступа, и комментарий к причине возврата в оборот.

Пример тела запроса:

```
{
  "card_id": 404,
  "reason": "Была найдена на территории склада."
}
```

5.6.4. Метод CardChanged

Подписка на событие изменение карт доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

При изменении карты доступа в системе, придет следующее сообщение:

```
{
  "card_id": 433,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- card_id – идентификатор карты доступа, с которой произошло изменение;
- change_type – тип изменения карты доступа (добавление, удаление, обновление или неизвестное).

5.7. DriverPassProfileService

Файл driver_pass_profile.proto предназначен для получения информации о профилях пропусков.

5.7.1. Метод GetDriverPassProfileTypes

Запрос на получение списка всех типов профилей пропусков установленных в системе. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "profile_types": [
    {
      "db_id": 1,
      "driver_id": 100,
      "internal_id": 3,

```

```
    "name": {
      "value": "профиль 1"
    },
    "pass_link_type": 0,
    "is_required": false
  },
  {
    "db_id": 2,
    "driver_id": 101,
    "internal_id": 3,
    "name": {
      "value": "профиль 2"
    },
    "pass_link_type": 0,
    "is_required": false
  },
  ...
]
```

- db_id – идентификатор профиля;
- driver_id – идентификатор типа драйвера;
- internal_id – внутренний для драйвера идентификатор типа профилей;
- name – наименование профиля;
- pass_link_type – тип связи пропуска к экземплярам данного типа профилей;
- is_required – должен ли в наборах пропусков обязательно быть выбран данный профиль.

5.7.2. Метод GetDriverPassProfiles

Запрос на получение всех добавленных в систему профилей пропусков для указанного типа профилей и типа драйвера. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "driver_id": 0,
  "profile_type_id": 2
}
```

- driver_id – идентификатор типа драйвера;
- profile_type_id – идентификатор типа профиля.

Пример ответа:

```
{
  "profiles": [
    {
      "profile_type_db_id": 2,
      "id": 2,
      "is_predefined": false,
      "name": "профиль 2",
      "description": {
```

```

        "value": "описание"
    },
    "is_default": false
},
{
    "profile_type_db_id": 2,
    "id": 4,
    "is_predefined": false,
    "name": "профиль 1",
    "description": {
        "value": "описание"
    },
    "is_default": false
},
...
]
```

- profile_type_db_id – идентификатор типа профиля настроек персонала;
- id – идентификатор профиля;
- is_predefined – признак предопределенного профиля;
- name – наименование профиля;
- description – описание профиля;
- is_default – используется ли данный профиль как значение по умолчанию.

5.7.3. Метод GetDriverPassProfileForPass

Запрос на получение списка всех назначенных для пропуска профилей пропусков. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_id": 124
}
```

- pass_id – идентификатор пропуска, профили которого нужно получить.

Ответ:

```
{
  "linked_profiles": [
    {
      "profile_id": 20,
      "identification_type": "PASS_IDENTIFICATION_TYPE_BY_PIN_CODE"
    },
    {
      "profile_id": 21,
      "identification_type": "PASS_IDENTIFICATION_TYPE_BY_CARD"
    },
    ...
  ]
}
```

```
}

```

- profile_id – идентификатор профиля драйвера;
- identification_type – тип идентификации профиля (по номеру карты, по ПИН-коду, по ПИН-коду и номеру карты).

5.8. ExternalIntegration

Файл external_integration.proto предназначен для отслеживания запросов на подтверждение доступа.

5.8.1. Метод ExternalAccessConfirmRequired

Запрос на подписку к внешней системе запросов подтверждения доступа. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса с идентификатором внешней системы, от которого будем получать запросы на подтверждения доступа:

```
{
  "external_id": {
    "value": "43"
  }
}
```

При появлении запроса на подтверждение доступа, получим следующее сообщение:

```
{
  "request_id": 3,
  "pass_id": 129,
  "pass_point_sdn": 345,
  "pass_direction": "PASS_DIRECTION_TYPE_IN",
  "external_id": {
    "value": "43"
  }
}
```

- request_id – идентификатор запроса, используемый при последующей отправке в систему подтверждения или отказа в доступе;
- pass_id – идентификатор пропуска, для которого выполняется запрос на подтверждение доступа;
- pass_point_id – идентификатор точки прохода, через которую предполагается доступ;
- pass_direction – направление прохода, в котором предполагается выполнение доступа. Возможные значения:
 - PASS_DIRECTION_TYPE_UNSPECIFIED – не указано;
 - PASS_DIRECTION_TYPE_IN – вход;
 - PASS_DIRECTION_TYPE_OUT – выход.

- `external_id` – идентификатор внешней системы, в которую отправляется запрос на подтверждение доступа.

5.8.2. Метод `ConfirmAccess`

Запрос на подтверждение доступа по ранее полученному идентификатору запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "request_id": 3
}
```

- `request_id` – идентификатор ранее полученного запроса на подтверждение доступа во внешней системе.

При успешном подтверждении доступа придет пустое ответное сообщение.

5.8.3. Метод `DenyAccess`

Запрос на отказ в доступе для ранее полученного идентификатора запроса с возможностью указать причину. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "reason": {
    "value": "ушел"
  },
  "request_id": 3
}
```

- `request_id` – идентификатор ранее полученного запроса на подтверждение доступа во внешней системе;
- `reason` – причина отказа доступа.

При успешном отказе доступа придет пустое ответное сообщение.

5.9. OrganizationStructureService

Файл `organization_structure.proto` предназначен для получения информации об организациях и департаментах.

5.9.1. Метод `GetOrganizationStructureNodes`

Запрос получение организаций и департаментов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответные сообщения будут содержать информацию об узле организации/департамента:

```
{
  "node": {
    "id": 2,
    "name": "Отдел не указан",
    "parent_id": {
      "value": 1
    },
    "node_type": "ORGANIZATION_NODE_TYPE_DEPARTMENT"
  }
}
```

- `id` – уникальный идентификатор организации/департамента;
- `name` – наименовании организации/департамента (ограничение 255 символов);
- `parent_id` – содержит значение уникального идентификатора родительской организации. Для корневой организации «Все» значение родительской организации равно *null* для остальных обязательно должно быть значение не равное *null*, указывающий на родительскую организацию;
- `node_type` – тип узла:
 - `ORGANIZATION_NODE_TYPE_UNSPECIFIED` – значение не определено;
 - `ORGANIZATION_NODE_TYPE_ORGANIZATION` – организация;
 - `ORGANIZATION_NODE_TYPE_DEPARTMENT` – департамент.

5.9.2. Метод `OrganizationStructureChanged`

Запрос отслеживания изменений дерева организаций/департаментов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "organization_node_id": 0,
  "change_type": "ENTITY_CHANGE_TYPE_UNSPECIFIED"
}
```

- `organization_node_id` – идентификатор организации/департамента, который изменился;
- `change_type` – тип изменения организации/департамента:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – запись создана;
 - `ENTITY_CHANGE_TYPE_UPDATE` – запись обновлена;
 - `ENTITY_CHANGE_TYPE_DELETE` – запись удалена.

5.10. `ControlAreaService`

Файл `control_area.proto` предназначен для получения информации о территориях.

5.10.1. Метод GetControlArea

Для получения информации о необходимой территории, требуется отправить запрос с указанием идентификатора территории. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "ctrl_area_id": 101
}
```

В поле `ctrl_area_id` нужно указать идентификатор территории, информацию которой желаете получить.

Ответное сообщение:

```
{
  "control_area": {
    "id": 101,
    "name": "Территория",
    "parent_id": {
      "value": 2
    },
    "time_block_id": {
      "value": 5
    }
  }
}
```

- `id` – уникальный идентификатор территории;
- `name` – наименовании территории;
- `parent_id` – содержит числовое значение идентификатора родительской территории. Если территория не принадлежит другой территории, то данное поле будет содержать значение `null`;
- `time_block_id` – идентификатор временного блока, которого принадлежит территория. Если у территории не указан временной блок, то данное поле содержит значение `null`.

При отправки с пустым телом запроса, ответное сообщение будет содержать территорию «Не указано».

5.10.2. Метод GetControlAreas

Запрос на получение списка всех территорий. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

```
{
  "control_areas": [
    {
      "id": 0,
      "name": "Не указано",
      "parent_id": null,
      "time_block_id": null
    },
  ],
}
```

```
{
  "id": 1,
  "name": "Вне территории",
  "parent_id": null,
  "time_block_id": null
},
...
]
```

5.10.3. Метод GetControlAreaPassPointLinks

Запрос на получение точек прохода с привязанными территориями. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример пакета сообщения:

```
{
  "control_area_pass_point_links": {
    "pass_point_sdn": 349,
    "source_area_id": 100,
    "destination_area_id": 1
  }
}
```

- pass_point_sdn – уникальный идентификатор точки прохода;
- source_area_id – идентификатор территории откуда ведет точка проходу;
- destination_area_id – идентификатор территории куда ведет точка прохода.

Точки прохода, которые изначально не были привязаны ни к одной территории, в данный пакет ответных сообщений не попадут.

5.10.4. Метод ControlAreaChanged

Запрос на отслеживание изменений территорий. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "control_area_id": 100,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- control_area_id – идентификатор территорий, которая изменилась;
- change_type – тип изменения карты доступа (добавление, удаление, обновление или неизвестное).

5.10.5. Метод ControlAreaPassPointLinkUpdated

Запрос на отслеживание изменений направлений точек прохода. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "pass_point_sdn": 351,
  "source_control_area_id": 1,
  "destination_control_area_id": 101
}
```

- pass_point_sdn – уникальный идентификатор точки прохода;
- source_area_id – идентификатор территории откуда ведет точка проходу;
- destination_area_id – идентификатор территории куда ведет точка прохода.

5.11. DictionariesService

Файл dictionaries.proto предназначен для работы со словарями и их записями.

5.11.1. Метод GetDictionaryHeaders

Запрос на получение списка словарей. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

```
{
  "dictionary_headers": [
    {
      "domains": [],
      "id": 2,
      "name": "Виды документов",
      "is_system": true
    },
    {
      "domains": [],
      "id": 3,
      "name": "Гражданство",
      "is_system": true
    },
    {
      "domains": [],
      "id": 101,
      "name": "Справочник 1",
      "is_system": false
    },
    ...
  ]
}
```

- domains – массив доменов, в которых допускается использование записей из справочника. Используется для пользовательских (is_system = false) справочников. Возможные значения:
 - DICTIONARY_DOMAINS_UNSPECIFIED – домен не определён;
 - DICTIONARY_DOMAINS_DEVICE_FIELDS – доступно использование в качестве значений дополнительных полей устройств;
 - DICTIONARY_DOMAINS_PASS_FIELDS – доступно использование в качестве значений дополнительных полей персональных пропусков.
- id – уникальный идентификатор словаря;
- name – наименование словаря;
- is_system – указание на то что словарь является системным.

5.11.2. Метод GetDictionaryRecords

Запрос на получение всех записей в словаре. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "header_id": 2
}
```

- header_id – идентификатор словаря.

Ответ содержит все записи указанного словаря:

```
{
  "dictionary_records": [
    {
      "id": 4,
      "header_id": 2,
      "value": "Паспорт",
      "is_system": false
    },
    {
      "id": 5,
      "header_id": 2,
      "value": "Удостоверение военнослужащего",
      "is_system": false
    }
  ]
}
```

- id – идентификатор записи;
- header_id – идентификатор принадлежности к словарю;
- value – значение записи (ограничение 255 символов);
- is_system – указание, является ли запись системной.

5.11.3. Метод GetDictionaryRecord

Запрос на получение записи словаря. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "id": 2
}
```

- id – идентификатор записи.

Ответ содержит информацию о записи в словаре:

```
{
  "dictionary_record": {
    "id": 5,
    "header_id": 2,
    "value": "Удостоверение военнослужащего",
    "is_system": false
  }
}
```

- id – идентификатор записи;
- header_id – идентификатор принадлежности к словарю;
- value – значение записи;
- is_system – указание, является ли запись системной.

5.11.4. Метод DictionaryRecordChanged

Запрос на отслеживание изменений в словарях. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "record_id": 78,
  "header_id": 5,
  "change_type": "ENTITY_CHANGE_TYPE_ADD",
  "record": {
    "id": 78,
    "header_id": 5,
    "value": "Удостоверение военнослужащего",
    "is_system": false
  }
}
```

- record_id – идентификатор записи;
- header_id – идентификатор принадлежности к словарю;

- `change_type` – тип изменения записи словаря:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – запись создана;
 - `ENTITY_CHANGE_TYPE_UPDATE` – запись обновлена;
 - `ENTITY_CHANGE_TYPE_DELETE` – запись удалена;
- `record` – изменённая словарная запись. При работе с сервером системы версии 2025.4 и выше, данное поле заполняется при получении событий о добавлении и обновлении записи справочника.

Так же можно указать в теле запроса идентификатор словаря, чтобы отслеживать изменения связанные только с указанным словарем:

```
{
  "header_id": {
    "value": 5
  }
}
```

5.11.5. Метод `GetDictionaryHeader`

Запрос на получение информации о справочнике по его идентификатору. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "header_id": 2
}
```

- `header_id` – идентификатор справочника.

Ответ содержит информацию справочнике:

```
{
  "dictionary_header": {
    "domains": [],
    "id": 2,
    "name": "Виды документов",
    "is_system": true
  }
}
```

- `domains` – массив доменов, в которых допускается использование записей из справочника. Используется для пользовательских (`is_system = false`) справочников. Возможные значения:
 - `DICTIONARY_DOMAINS_UNSPECIFIED` – домен не указан;
 - `DICTIONARY_DOMAINS_DEVICE_FIELDS` – доступно использование в качестве значений дополнительных полей устройств;

- `DICTIONARY_DOMAINS_PASS_FIELDS` – доступно использование в качестве значений дополнительных полей персональных пропусков.
- `id` – уникальный идентификатор словаря;
- `name` – наименование словаря;
- `is_system` – указание на то что словарь является системным.

5.11.6. Метод `DictionaryHeaderChanged`

Запрос на отслеживание изменений в списке справочников. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "header_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD",
  "header": {
    "domains": [],
    "id": 101,
    "name": "Справочник 1",
    "is_system": false
  }
}
```

- `header_id` – идентификатор изменившегося справочника;
- `change_type` – тип изменения справочника:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – справочник создан;
 - `ENTITY_CHANGE_TYPE_UPDATE` – справочник обновлён;
 - `ENTITY_CHANGE_TYPE_DELETE` – справочник удалён.
- `header` – заголовок изменённого справочника. При работе с сервером системы версии 2025.4 и выше, данное поле заполняется при получении событий о добавлении и обновлении справочника.

5.12. PassService

Файл `pass.proto` предназначен для работы с пропусками персон. Подробнее о пропусках в документации «Бастион-3 – Бюро пропусков. Руководство оператора».

5.12.1. Метод `GetPass`

Запрос на получение информации о пропуске. Запрос может быть выполнен только с использованием токена доступа.

Запрос:

```
{
```

```
"pass_id": 147
}
```

ОТВЕТ:

```
{
  "pass": {
    "id": 147,
    "person_id": 142,
    "pass_category_id": 1,
    "status": "PASS_STATUS_RETURNED",
    "card_id": {
      "value": 413
    },
    "access_level_id": {
      "value": 121
    },
    "priority": 1,
    "create_date": {
      "seconds": "1712906231",
      "nanos": 595189000
    },
    "activation_time": null,
    "issue_date": {
      "seconds": "1712906246",
      "nanos": 799649000
    },
    "start_date": null,
    "end_date": null,
    "visit_goal_id": null,
    "confirm_person_id": null,
    "return_reason_id": {
      "value": 19
    },
    "return_date": {
      "seconds": "1712906691",
      "nanos": 877843000
    },
    "disable_anti_pass_back_check": false,
    "accept_person_id": null,
    "accept_department_id": null,
    "block_reason_id": null,
    "blocked_date": null,
    "unblocked_date": null,
    "comment": null,
    "number": null
  }
}
```

- id – уникальный идентификатор пропуска;
- person_id – идентификатор привязанной персоны;
- pass_category_id – идентификатор категории пропуска;
- status – статус пропуска (полный список статусов можно посмотреть в файле common.proto);
- card_id – числовое значение идентификатора карты доступа, привязанный к пропуску;
- access_level_id – числовое значение идентификатора уровня доступа;

- priority – приоритет пропуска;
- create_date – дата создания пропуска;
- activation_time – дата активации пропуска;
- issue_date – дата выдачи пропуска.
- start_date – дата начала действия пропуска;
- end_date – дата окончания действия пропуска;
- visit_goal_id – идентификатор значения словаря «Цель посещения»;
- confirm_person_id – идентификатор персоны принявший заявку;
- return_reason_id – идентификатор значения словаря «Причина возврата пропуска»;
- return_date – дата возврата пропуска;
- disable_anti_pass_back_check – указатель, может ли пропуск быть повторно выдан;
- accept_person_id – идентификатор принимающего лица;
- accept_department_id – идентификатор принимающего подразделения;
- block_reason_id – идентификатор значения словаря «Причина блокировки»;
- blocked_date – дата блокировки пропуска;
- unblocked_date – дата разблокировки пропуска;
- comment – комментарий к пропуску (ограничение 100 символов);
- number – номер пропуска (ограничение 16 символов).

5.12.2. Метод GetPasses

Если необходимо получить информацию не об одном пропуске, а о нескольких, можно использовать следующий запрос, где необходимо перечислить идентификаторы искомых пропусков:

```
{
  "pass_ids": [
    147,
    110,
    109
  ]
}
```

Запрос может быть выполнен только с использованием токена доступа.

В качестве результата получаем список пропусков:

```
{
  "passes": [
    {
      "id": 109,
      "person_id": 106,
      "pass_category_id": 1,
      "status": "PASS_STATUS_ACTIVE",
      "card_id": {
        "value": 261
      },
      "access_level_id": {
```

```

        "value": 121
    },
    "priority": 1,
    "create_date": {
        "seconds": "1712223312",
        "nanos": 820706000
    },
    "activation_time": null,
    "issue_date": {
        "seconds": "1712904199",
        "nanos": 906148000
    },
    "start_date": null,
    "end_date": null,
    "visit_goal_id": null,
    "confirm_person_id": null,
    "return_reason_id": null,
    "return_date": null,
    "disable_anti_pass_back_check": false,
    "accept_person_id": null,
    "accept_department_id": null,
    "block_reason_id": null,
    "blocked_date": null,
    "unblocked_date": null,
    "comment": null
    "number": null
    },
    ...
]
}

```

5.12.3. Метод GetPassesForPerson

Запрос на получение всех пропусков у персоны. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "person_id": 107
}

```

- person_id – идентификатор персоны, пропуска которого необходимо получить.

В ответе получаем список аналогично запросу GetPasses, но в список входят только пропуска, которые относятся к указанной персоне.

5.12.4. Метод GetPassPinCode

Запрос на получение ПИН-кода пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_id": 110
}
```

- pass_id – идентификатор пропуска, ПИН-код которого необходимо получить.

Ответ приходит с числовым значением ПИН-кода:

```
{
  "pin_code": {
    "value": "0001"
  }
}
```

5.12.5. Метод GetPassPinCodes

Запрос на получение списка ПИН-кодов у пропусков. Запрос может быть выполнен только с использованием токена доступа.

В запросе необходимо перечислить все идентификаторы пропусков, ПИН-кодов которых требуется получить:

```
{
  {
    "pass_ids": [
      124,
      110,
      143,
      144,
      109
    ]
  }
}
```

Ответ будет содержать список указывающий, какой ПИН-код принадлежит какому пропуску:

```
{
  "pin_code_infos": [
    {
      "pass_id": 109,
      "pin_code": "0572"
    },
    {
      "pass_id": 110,
      "pin_code": "9801"
    }
  ]
}
```

- pass_id – идентификатор пропуска, которому принадлежит ПИН-код;

- pin_code – ПИН-код пропуска.

Если в запросе указаны идентификаторы пропусков, у которых нет ПИН-кода, то в ответном сообщении информации о них не будет.

5.12.6. Метод GetCardReplacementInfo

Запрос на получение списка информации о замене карты доступа. Запрос может быть выполнен только с использованием токена доступа.

Для запроса необходимо указать идентификатор пропуска, информацию о которой необходимо получить:

```
{
  "pass_id": 110
}
```

Ответное сообщение содержит список карты замены:

```
{
  "card_replacements": [
    {
      "id": 1,
      "pass_id": 142,
      "previous_card_id": 333,
      "replacement_date": {
        "seconds": "1713093000",
        "nanos": 957140000
      },
      "reason": "CARD_REPLACEMENT_REASON_RETURN",
      "return_reason_id": {
        "value": 120
      }
    },
    {
      "id": 1,
      "pass_id": 142,
      "previous_card_id": 333,
      "replacement_date": {
        "seconds": "1713093000",
        "nanos": 957140000
      },
      "reason": "CARD_REPLACEMENT_REASON_WITHDRAW",
      "withdrawal_reason": "PASS_WITHDRAWAL_REASON_UNUSABLE"
    },
    ...
  ]
}
```

- id – идентификатор карты возврата;
- pass_id – идентификатор пропуска;
- previous_card_id – идентификатор предыдущей карты доступа;

- `reason` – причина возврата. Карта доступа может быть либо возвращена `CARD_REPLACEMENT_REASON_RETURN`, либо изъята `CARD_REPLACEMENT_REASON_WITHDRAW`;
- `return_reason_id` – идентификатор записи словаря «Причины возврата пропусков». Не указывается, если карта была изъята;
- `withdrawal_reason` – причина изъятия пропуска. Не указывается, если карта была возвращена.

5.12.7. Метод `GetPassSecurityControlGroupSettings`

Запрос на получение информации об управлении охраной для пропуска. Запрос может быть выполнен только с использованием токена доступа.

Для запроса необходимо указать идентификатор пропуска, информацию о которой необходимо получить:

```
{
  "pass_id": 110
}
```

Ответное сообщение содержит информацию об управлении охраной указанного пропуска:

```
{
  "setting": {
    "pass_id": 149,
    "ops_group_id": 2,
    "pass_identification_type": "PASS_IDENTIFICATION_TYPE_BY_PIN_CODE"
  }
}
```

- `pass_id` – идентификатор пропуска;
- `ops_group_id` – идентификатор группы управления охраной;
- `pass_identification_type` – тип идентификации пропуска (по номеру карты, по ПИН-коду, по ПИН-коду и номеру карты).

Если к пропуска не привязано управление охраной, то результатом запроса будет:

```
{
  "setting": null
}
```

5.12.8. Метод `IssuePass`

Запрос на выдачу пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "card_id": 373,
```

```
"issue_date": {
  "nanos": 728198743,
  "seconds": 542916
},
"pass_id": 150
}
```

- pass_id – идентификатор пропуска, который нужно выдать;
- card_id – идентификатор карты доступа для выдачи;
- issue_date – дата выдачи пропуска. Этот параметр является необязательным. Если он не задан, тогда используется текущее время сервера системы.

Если выдача прошла успешно, то придет сообщение с пустым телом.

5.12.9. Метод ReturnPass

Запрос на возврат пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_id": 110,
  "return_reason_id": {
    "value": 5
  }
}
```

- pass_id – идентификатор пропуска, который нужно вернуть;
- return_reason_id – идентификатор значения словаря «Причины возврата пропусков».

Если возврат прошел успешно, то придет сообщение с пустым телом.

5.12.10. Метод WithdrawPass

Запрос на изъятия пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_id": 142,
  "reason": "PASS_WITHDRAWAL_REASON_CHARGED_OFF"
}
```

- pass_id – идентификатор пропуска, который нужно изъять;
- reason – причина изъятия:
 - PASS_WITHDRAWAL_REASON_UNSPECIFIED – не указанная причина;
 - PASS_WITHDRAWAL_REASON_UNUSABLE – пропуск пришел в негодность;
 - PASS_WITHDRAWAL_REASON_CHARGED_OFF – пропуск списан;

- PASS_WITHDRAWAL_REASON_LOST – пропуск утерян.

Если изъятие прошло успешно, то придет сообщение с пустым телом.

5.12.11. Метод ProlongPass

Запрос на продление пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "new_end_time": {
    "nanos": 244543620,
    "seconds": 369878714
  },
  "pass_id": 138,
  "prolong_reason": {
    "value": "Не указано"
  }
}
```

- pass_id – идентификатор пропуска, который нужно продлить;
- new_end_time – новая дата окончания действия пропуска;
- prolong_reason – причина продления пропуска.

Если продление прошло успешно, то придет сообщение с пустым телом.

5.12.12. Метод ReplaceCardWithReturn

Запрос на замену карты доступа, в качестве по причине возврата карты. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "new_card_id": 2,
  "pass_id": 110,
  "return_reason_id": 5
}
```

- pass_id – идентификатор пропуска, карту которого необходимо заменить;
- new_card_id – идентификатор карты доступа, на которую надо заменить;
- return_reason_id – идентификатор значение словаря «Причины возврата пропуска».

Если замена прошла успешно, то придет сообщение с пустым телом.

5.12.13. Метод ReplaceCardWithWithdraw

Запрос на замену карты доступа, причина которого изъятие. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "new_card_id": 2,
  "pass_id": 110,
  "withdraw_reason": "PASS_WITHDRAWAL_REASON_UNUSABLE"
}
```

- pass_id – идентификатор пропуска, карту которого необходимо заменить;
- new_card_id – идентификатор карты доступа, на которую надо заменить;
- withdraw_reason – причина изъятия:
 - PASS_WITHDRAWAL_REASON_UNSPECIFIED – не указанная причина;
 - PASS_WITHDRAWAL_REASON_UNUSABLE – пропуск пришел в негодность;
 - PASS_WITHDRAWAL_REASON_CHARGED_OFF – пропуск списан;
 - PASS_WITHDRAWAL_REASON_LOST – пропуск утерян.

5.12.14. Метод PassChanged

Запрос на отслеживание изменений в пропусках. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответного сообщения от события изменения пропуска:

```
{
  "pass_id": 150,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- pass_id – идентификатор пропуска, который изменился;
- change_type – тип изменения пропуска:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – пропуск создан;
 - ENTITY_CHANGE_TYPE_UPDATE – пропуск обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – пропуск удален.

5.13. PassCategoryService

Файл pass_category.proto предназначен для управления категориями пропусков.

5.13.1. Метод GetPassCategory

Запрос на получение информации о категории пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_category_id": 2
}
```

- `pass_category_id` – идентификатор категории пропуска, информацию о которой необходимо получить.

Ответ:

```
{
  "pass_category": {
    "id": 2,
    "name": "Контрагенты",
    "time_restriction_rule": "TIME_RESTRICTION_RULE_END",
    "time_restriction_unit": "TIME_RESTRICTION_UNIT_YEAR",
    "time_restriction_value": 1,
    "photo_identification_form_id": {
      "value": 2
    },
    "is_change_pass_end_date_allowed": true,
    "is_pass_prolongation_allowed": true,
    "numeration_id": {
      "value": 3
    },
    "inactive_days_count_before_auto_block_pass": {
      "value": 2
    },
    "activation_time_limit": {
      "value": 2
    }
  }
}
```

- `id` – идентификатор категории пропуска;
- `name` – наименование категории;
- `time_restriction_rule` – правило ограничения срока действия для пропусков, входящих в категорию:
 - `TIME_RESTRICTION_RULE_UNSPECIFIED` – не указан;
 - `TIME_RESTRICTION_RULE_NONE` – нет ограничения по сроку действия пропуска;
 - `TIME_RESTRICTION_RULE_FULL` – срок действия пропуска ограничивается заданное количество временных интервалов с момента начала действия пропуска;
 - `TIME_RESTRICTION_RULE_END` – срок действия пропуска ограничивается до конца временного интервала, в который попадает начало действия пропуска;

- `time_restriction_uint` – тип интервала времени, применяемого при расчёте срока действия для пропусков, входящих в категорию:
 - `TIME_RESTRICTION_UNIT_UNSPECIFIED` – не указан;
 - `TIME_RESTRICTION_UNIT_DAY` – день;
 - `TIME_RESTRICTION_UNIT_WEEK` – неделя;
 - `TIME_RESTRICTION_UNIT_MONTH` – месяц;
 - `TIME_RESTRICTION_UNIT_YEAR` – год;
- `time_restriction_value` – количество интервалов времени применяемое при расчёте срока действия для пропусков, входящих в категорию;
- `photo_identification_form_id` – идентификатор формы фото идентификации, отображаемой для пропусков, входящих в категорию;
- `is_change_pass_end_date_allowed` – признак возможности ручного изменения даты окончания действия для пропусков, входящих в категорию;
- `is_pass_prolongation_allowed` – признак возможности пролонгации срока действия для пропусков, входящих в категорию;
- `numeration_id` – идентификатор нумерации пропусков;
- `inactive_days_count_before_auto_block_pass` – количество дней отсутствия активности пропуска до его автоматической блокировки;
- `activation_time_limit` – ограничение на время активации пропусков.

5.13.2. Метод `GetPassCategories`

Запрос на получения списка всех категорий пропусков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "pass_categories": [
    {
      "id": 1,
      "name": "Сотрудники",
      "time_restriction_rule": "TIME_RESTRICTION_RULE_NONE",
      "time_restriction_unit": "TIME_RESTRICTION_UNIT_DAY",
      "time_restriction_value": 1,
      "photo_identification_form_id": {
        "value": 1
      },
      "is_change_pass_end_date_allowed": true,
      "is_pass_prolongation_allowed": true,
      "numeration_id": null,
      "inactive_days_count_before_auto_block_pass": null,
      "activation_time_limit": null
    },
    {
      "id": 3,
      "name": "Посетители",
      "time_restriction_rule": "TIME_RESTRICTION_RULE_END",
      "time_restriction_unit": "TIME_RESTRICTION_UNIT_DAY",
      "time_restriction_value": 1,
      "photo_identification_form_id": {
```

```

        "value": 2
    },
    "is_change_pass_end_date_allowed": false,
    "is_pass_prolongation_allowed": false,
    "numeration_id": null,
    "inactive_days_count_before_auto_block_pass": null,
    "activation_time_limit": null
  },
  ...
]
}

```

5.13.3. Метод `GetPassCategoriesAvailabilityInfo`

Запрос на получения списка активных подразделений для категорий. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```

{
  "availability_info": [
    {
      "organization_node_ids": [
        0,
        104
      ],
      "pass_category_id": 1
    },
    {
      "organization_node_ids": [
        0,
        101,
        102,
        103,
        104,
        105
      ],
      "pass_category_id": 2
    },
    {
      "organization_node_ids": [
        100,
        103
      ],
      "pass_category_id": 3
    },
    ...
  ]
}

```

Availability_info содержит список идентификаторов доступных организаций *organization_node_ids*, для определенной категории *pass_category_id*.

5.13.4. Метод `GetOrganizationAllowedPassCategories`

Запрос на получение привязанных категорий пропуска к подразделению. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "organization_node_id": 100
}
```

- `organization_node_id` – идентификатор подразделения;

Ответ содержит список идентификаторов категорий пропусков, которые доступны на указанном подразделении:

```
{
  "pass_category_ids": [
    3,
    1
  ]
}
```

5.13.5. Метод `PassCategoryChanged`

Запрос на подписку событий изменения категорий пропусков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "pass_category_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- `pass_category_id` – идентификатор категории пропуска, которая изменилась;
- `change_type` – тип изменения пропуска:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – категория создана;
 - `ENTITY_CHANGE_TYPE_UPDATE` – категория обновлена;
 - `ENTITY_CHANGE_TYPE_DELETE` – категория удалена.

5.14. `PersonService`

Файл `person.proto` предназначен для работы с персонами.

5.14.1. Метод GetPerson

Запрос на получение информации о персоне. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_id": 107
}
```

- person_id – идентификатор персоны.

Пример ответа:

```
{
  "person": {
    "id": 107,
    "name": "Иванов",
    "first_name": {
      "value": "Иван"
    },
    "second_name": {
      "value": "Иванович"
    },
    "table_no": {
      "value": "aug65"
    },
    "comments": {
      "value": "Работает только до августа"
    },
    "organization_node_id": 0,
    "position_id": {
      "value": 18
    },
    "add_field_1": null,
    "add_field_2": null,
    "add_field_3": null,
    "add_field_4": {
      "value": "Женат"
    },
    "add_field_5": {
      "value": "2 детей"
    },
    "add_field_6": null,
    "add_field_7": null,
    "add_field_8": null,
    "add_field_9": null,
    "add_field_10": null,
    "add_field_11": null,
    "add_field_12": null,
    "add_field_13": null,
    "add_field_14": null,
    "add_field_15": null,
    "add_field_16": null,
    "add_field_17": null,
    "add_field_18": null,
  }
}
```

```
    "add_field_19": null,
    "add_field_20": null,
    "create_date": {
      "seconds": "1712227017",
      "nanos": 748900000
    }
  }
}
```

- id – идентификатор персоны;
- name – фамилия персоны (ограничение 100 символов);
- first_name – имя персоны (ограничение 100 символов);
- second_name – отчество персоны (ограничение 100 символов);
- table_no – табельный номер персоны (ограничение 20 символов);
- comments – комментарий к персоне (ограничение 2000 символов);
- organization_node_id – идентификатор подразделения, к которой принадлежит персона;
- position_id – идентификатор должности персоны;
- add_field[1..20] – значение дополнительных полей (ограничение 255 символов);
- create_date – дата создания персоны.

5.14.2. Метод GetPersonAdditionalFields

Запрос на получение списка настроек дополнительных полей для персон. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "additional_fields": [
    {
      "number": 1,
      "name": "Нарушения",
      "is_used": true
    },
    {
      "number": 2,
      "name": "Нарушения -2-е",
      "is_used": true
    },
    {
      "number": 3,
      "name": "Нарушения -3-е",
      "is_used": true
    },
    {
      "number": 4,
      "name": "Задер с приз-ми алкогольного опьянения",
      "is_used": true
    },
    {
      "number": 5,
      "name": "Задер с приз-ми алкогольного опьянения",
      "is_used": true
    }
  ]
}
```

```
    },
    {
      "number": 6,
      "name": "Вынос",
      "is_used": true
    },
    {
      "number": 7,
      "name": "Вывоз",
      "is_used": true
    },
    {
      "number": 8,
      "name": "Попытка проноса на территорию",
      "is_used": true
    },
    {
      "number": 9,
      "name": "Нарушения 4-е",
      "is_used": true
    },
    {
      "number": 10,
      "name": "Нарушения 5-е",
      "is_used": true
    },
    {
      "number": 11,
      "name": "6",
      "is_used": false
    },
    {
      "number": 12,
      "name": "7",
      "is_used": false
    },
    {
      "number": 13,
      "name": "8",
      "is_used": false
    },
    {
      "number": 14,
      "name": "Дополнительное поле 14",
      "is_used": false
    },
    {
      "number": 15,
      "name": "Дополнительное поле 15",
      "is_used": false
    },
    {
      "number": 16,
      "name": "Дополнительное поле 16",
      "is_used": false
    },
    {
      "number": 17,
      "name": "Дополнительное поле 17",
      "is_used": true
    },
  ],
```

```

    {
      "number": 18,
      "name": "Дополнительное поле 18",
      "is_used": false
    },
    {
      "number": 19,
      "name": "Дополнительное поле 19",
      "is_used": false
    },
    {
      "number": 20,
      "name": "Проход",
      "is_used": true
    }
  ]
}

```

- number – номер дополнительного поля;
- name – наименование дополнительного поля (ограничение 100 символов);
- is_used – признак использования дополнительного поля.

5.14.3. Метод GetPersonInfo

Запрос на получение персональных данных о персоне. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "person_id": 107
}

```

- person_id – идентификатор персоны.

Ответ:

```

{
  "personal_info": {
    "person_id": 107,
    "birth_place": {
      "value": "Москва"
    },
    "birth_date": {
      "year": 2024,
      "month": 4,
      "day": 4
    },
    "document_type_id": {
      "value": 4
    },
    "document_number": {
      "value": "2222222"
    }
  },

```

```
    "document_series": {
      "value": "1111111"
    },
    "document_issue_organ": {
      "value": 9
    },
    "document_issue_date": {
      "year": 2024,
      "month": 4,
      "day": 3
    },
    "phone": {
      "value": "888888888888"
    },
    "address": {
      "value": "Москва"
    },
    "citizenship_id": {
      "value": 3
    },
    "gender": "GENDER_MALE",
    "email": {
      "value": "email@email"
    },
    "additional_phone": null
  }
}
```

- person_id – идентификатор персоны;
- birth_place – место рождения (ограничение 60 символов);
- birth_date – дата рождения;
- document_type_id – идентификатор значения словаря «Виды документов»;
- document_number – номер документа (ограничение 12 символов);
- document_series – серия документа (ограничение 12 символов);
- document_issue_organ – орган выдавший документ;
- document_issue_date – дата выдачи документа;
- phone – телефон (ограничение 15 символов);
- address – адрес проживания (ограничение 255 символов);
- citizenship_id – идентификатор значения словаря «Гражданство»;
- gender – пол;
- email – электронная почта (ограничение 70 символов);
- additional_phone – дополнительный телефон (ограничение 15 символов).

Данные поля могут быть не заполнены:

```
{
  "personal_info": {
    "person_id": 107,
    "birth_place": null,
    "birth_date": null,
    "document_type_id": null,
    "document_number": null,

```

```
    "document_series": null,  
    "document_issue_organ": null,  
    "document_issue_date": null,  
    "phone": null,  
    "address": null,  
    "citizenship_id": null,  
    "gender": "GENDER_UNSPECIFIED",  
    "email": null,  
    "additional_phone": null  
  }  
}
```

5.14.4. Метод GetPersonalInfos

Запрос на получение списка персональных данных персон. Запрос может быть выполнен только с использованием токена доступа.

В запросе необходимо указать список идентификаторов персон, о которых требуется получить персональные данные:

```
{  
  "person_ids": [  
    107,  
    144,  
    101  
  ]  
}
```

Ответ содержит список с персональными данными о найденных персонах:

```
{  
  "personal_infos": [  
    {  
      "person_id": 107,  
      "birth_place": {  
        "value": "Москва"  
      },  
      "birth_date": {  
        "year": 2024,  
        "month": 3,  
        "day": 27  
      },  
      "document_type_id": {  
        "value": 4  
      },  
      "document_number": null,  
      "document_series": null,  
      "document_issue_organ": {  
        "value": 9  
      },  
      "document_issue_date": {  
        "year": 2024,  
        "month": 4,  
        "day": 8  
      },  
    },  
  ],  
}
```

```

        "phone": {
            "value": "888888"
        },
        "address": {
            "value": "Москва"
        },
        "citizenship_id": {
            "value": 3
        },
        "gender": "GENDER_MALE",
        "email": {
            "value": "email@email"
        },
        "additional_phone": null
    },
    {
        "person_id": 144,
        "birth_place": null,
        "birth_date": null,
        "document_type_id": null,
        "document_number": null,
        "document_series": null,
        "document_issue_organ": null,
        "document_issue_date": null,
        "phone": null,
        "address": null,
        "citizenship_id": null,
        "gender": "GENDER_UNSPECIFIED",
        "email": null,
        "additional_phone": null
    },
    ...
]
}

```

5.14.5. Метод GetPersonalPhoto

Запрос для получения фото персоны. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "person_id": 107
}

```

- person_id – идентификатор персоны.

Ответ содержит значение фотографии:

```

{
  "photo": {
    "value": "QEBAQEBAQEBAQEBAQE..."
  }
}

```

```
}  
}
```

Если фото у персоны не задана придет значение null:

```
{  
  "photo": null  
}
```

5.14.6. Метод GetPersonPhotoHash

Запрос на получение хеша фотографии персоны. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{  
  "person_id": 107  
}
```

- person_id – идентификатор персоны.

Ответ содержит значение хеша фотографии:

```
{  
  "hash": {  
    "value": "b3ed4a19a4bc556ebffe680e1cdddd61"  
  }  
}
```

Если фото у персоны не задана придет значение null:

```
{  
  "hash": null  
}
```

5.14.7. Метод GetPersonPhotoThumbnail

Запрос на получение превью фотографии. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{  
  "person_id": 107  
}
```

- person_id – идентификатор персоны.

Ответ содержит значение хеша фотографии:

```
{
  "thumbnail": {
    "value": "/9j/4AAQSkZJRgABAQAAQABAAD/2wCEAAEBAQE..."
  }
}
```

Если фото у персоны не задана придет значение null:

```
{
  "thumbnail": null
}
```

5.14.8. Метод GetPersons

Запрос на получение списка персон. Запрос может быть выполнен только с использованием токена доступа.

Запрос формируется из списка идентификаторов персон, информацию о которых необходимо получить:

```
{
  "person_ids": [
    139,
    144,
    80,
    101
  ]
}
```

Ответ содержит список только тех персон, которые были найдены по указанным идентификаторам:

```
{
  "persons": [
    {
      "id": 139,
      "name": "Иван",
      "first_name": null,
      "second_name": null,
      "table_no": null,
      "comments": null,
      "organization_node_id": 0,
      "position_id": null,
      "add_field_1": null,
      "add_field_2": null,
      "add_field_3": null,
      "add_field_4": {
        "value": "Холост"
      }
    }
  ]
}
```

```

    },
    "add_field_5": {
      "value": ""
    },
    "add_field_6": null,
    "add_field_7": null,
    "add_field_8": null,
    "add_field_9": null,
    "add_field_10": null,
    "add_field_11": null,
    "add_field_12": null,
    "add_field_13": null,
    "add_field_14": null,
    "add_field_15": null,
    "add_field_16": null,
    "add_field_17": null,
    "add_field_18": null,
    "add_field_19": null,
    "add_field_20": null,
    "create_date": {
      "seconds": "1712853484",
      "nanos": 103671000
    }
  },
  {
    "id": 144,
    "name": "Дмитрий",
    "first_name": null,
    "second_name": null,
    "table_no": null,
    "comments": null,
    "organization_node_id": 0,
    "position_id": null,
    "add_field_1": null,
    "add_field_2": null,
    "add_field_3": null,
    "add_field_4": null,
    "add_field_5": null,
    "add_field_6": null,
    "add_field_7": null,
    "add_field_8": null,
    "add_field_9": null,
    "add_field_10": null,
    "add_field_11": null,
    "add_field_12": null,
    "add_field_13": null,
    "add_field_14": null,
    "add_field_15": null,
    "add_field_16": null,
    "add_field_17": null,
    "add_field_18": null,
    "add_field_19": null,
    "add_field_20": null,
    "create_date": {
      "seconds": "1712906764",
      "nanos": 685790000
    }
  }
]
}

```

Если ни одна персона не была найдена по идентификаторам, в ответ придет пустой список:

```
{
  "persons": []
}
```

5.14.9. Метод `PersonAdditionalFieldsChanged`

Запрос на получение уведомлений об изменениях настроек дополнительных полей для персон. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

5.14.10. Метод `PersonChanged`

Запрос на подписку изменения персон. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "person_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- `person_id` – идентификатор персоны, которая изменилась;
- `change_type` – тип изменения персоны:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – персона создана;
 - `ENTITY_CHANGE_TYPE_UPDATE` – персона обновлена;
 - `ENTITY_CHANGE_TYPE_DELETE` – персона удалена.

5.14.11. Метод `PersonInfoChanged`

Запрос на подписку изменения персональных данных персон. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "person_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- `person_id` – идентификатор персоны, персональные данные которые изменились;
- `change_type` – тип изменения:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;

- ENTITY_CHANGE_TYPE_ADD – создано;
- ENTITY_CHANGE_TYPE_UPDATE – обновлено;
- ENTITY_CHANGE_TYPE_DELETE – удалено.

5.14.12. Метод PersonPhotoChanged

Запрос на подписку изменения фото персоны. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "person_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- person_id – идентификатор персоны, фото которой изменилось;
- change_type – тип изменения фото:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – фото создано;
 - ENTITY_CHANGE_TYPE_UPDATE – фото обновлено;
 - ENTITY_CHANGE_TYPE_DELETE – фото удалено.

5.15. SearchPassService

Файл search_pass.proto предназначен для поиска пропусков.

5.15.1. SearchPasses

Запрос на поиск пропусков с определенными критериями. Запрос может быть выполнен только с использованием токена доступа.

При поиске пропусков в параметрах запроса (SearchPassesRequest) можно указать дополнительные условия поиска (в поле terms), при этом сами условия поиска запаковываются в тип Any (см. [Приложение 1. Тип google.protobuf.Any](#)).

При использовании соответствующих фильтров, результатом будут сообщения с информацией о пропусках с указанными значениями фильтров подробнее о пропусках читайте в [соответствующем разделе](#):

```
{
  "pass": {
    "id": 142,
    "person_id": 107,
    "pass_category_id": 1,
    "status": "PASS_STATUS_CHARGED_OFF",
    "card_id": {
      "value": 493
    }
  },
}
```

```

    "access_level_id": null,
    "priority": 1,
    "create_date": {
      "seconds": "1712853460",
      "nanos": 148463000
    },
    "activation_time": null,
    "issue_date": {
      "seconds": "1712853851",
      "nanos": 446661000
    },
    "start_date": null,
    "end_date": null,
    "visit_goal_id": null,
    "confirm_person_id": null,
    "return_reason_id": null,
    "return_date": {
      "seconds": "1713094974",
      "nanos": 420667000
    },
    "disable_anti_pass_back_check": false,
    "accept_person_id": null,
    "accept_department_id": null,
    "block_reason_id": {
      "value": -3
    },
    "blocked_date": {
      "seconds": "1713110412",
      "nanos": 571347000
    },
    "unblocked_date": {
      "seconds": "1713110332",
      "nanos": 996133000
    },
    "comment": {
      "value": ""
    },
    "number": null
  }
}

```

5.15.2. Фильтры пропусков

Фильтры пропусков позволяют осуществлять поиск по основным параметрам (ФИО, дата создания, должность и т.д.). Параметры фильтра могут быть заданы с использованием стандартных элементов фильтрации, подробное описание которых приведено в [Приложении 3. Основные элементы фильтрации](#).

Поиск по атрибутам персон-владельцев пропусков.

type_url: type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByPersonAttributesSearchTerm.

Параметры фильтра:

- position – фильтр по должности персоны (IdsValueSearchTermEntry);
- organization_node_ids – фильтр по указанным идентификаторам подразделения (IdsSearchTermEntry);
- create_date – фильтр по дате создания пропуска (TimestampSearchTermEntry);

- name – фильтр по фамилии персоны (StringSearchTermEntry);
- first_name – фильтр по имени персоны (StringSearchTermEntry);
- second_name – фильтр по отчеству персоны (StringSearchTermEntry);
- table_no – фильтр по табельному номеру персоны (StringSearchTermEntry);
- comments – фильтр по комментарию к персоне (StringSearchTermEntry);
- has_photo – фильтр по наличию/отсутствию фотографии у персоны (google.protobuf.BoolValue).

Поиск по подразделению персон-владельцев пропусков.

type_url: type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByOrganizationSearchTerm.

Параметры фильтра:

- organization_node_id – идентификатор подразделения для поиска;
- include_nested_nodes – включать ли вложенные подразделения.

Поиск по основным атрибутам пропуска.

type_url: type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByPassAttributesSearchTerm.

Параметры фильтра:

- priority – фильтр по приоритету пропуска (Int32SearchTermEntry);
- statuses – фильтр по включенным статусам (PassStatusesSearchTermEntry):
 - statuses – список статусов для включения в поиск. Возможные статусы пропусков:
 - PASS_STATUS_UNSPECIFIED – не указан;
 - PASS_STATUS_NOT_ACTIVE – не активен;
 - PASS_STATUS_ACTIVE – активен;
 - PASS_STATUS_EXPIRED – срок действия истек;
 - PASS_STATUS_RETURNED – возвращен;
 - PASS_STATUS_UNUSABLE – пришел в негодность;
 - PASS_STATUS_CHARGED_OFF – списан;
 - PASS_STATUS_LOST – утерян;
 - PASS_STATUS_DENIAL – отказ в утверждении;
 - PASS_STATUS_NOT_CONFIRM – принят в утверждении;
- pass_category_ids – фильтр по категориям пропуска (IdsSearchTermEntry);
- create_date – фильтр по дате создания пропуска (TimestampSearchTermEntry);
- issue_date – фильтр по дате выдачи пропуска (TimestampValueSearchTermEntry);
- activation_date – фильтр по дате активации (первый проход) пропуска (TimestampValueSearchTermEntry);
- start_date – фильтр по дате начала действия пропуска (TimestampValueSearchTermEntry);
- end_date – фильтр по дате окончания действия пропуска (TimestampValueSearchTermEntry);
- blocked_date – фильтр по дате блокировки пропуска (TimestampValueSearchTermEntry);
- unblocked_date – фильтр по дате разблокирования пропуска (TimestampValueSearchTermEntry);
- return_date – фильтр по дате возврата пропуска (TimestampValueSearchTermEntry);
- last_passage_date – фильтр по дате последнего использования пропуска (TimestampValueSearchTermEntry);

- `return_reason` – фильтр по причинам возврата (`IdsValueSearchTermEntry`);
- `block_reason` – фильтр по причинам блокировки (`IdsValueSearchTermEntry`).

Поиск по вхождению заданных подстрок в основные параметры: ФИО, табельный номер, номер пропуска, код карты, комментарии к персоне и пропуску. (Для поиска быстрого поиска пропусков по табельному номеру рекомендуется использовать фильтр `PassByPersonAttributesSearchTerm`).

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByCommonPropertiesContainWordsSearchTerm`.

Параметры фильтра:

- `words` – фильтр ключевые слова для поиска пропусков (`repeated string`).

Поиск пропусков по правам доступа.

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByAccessPermissionsSearchTerm`.

Параметры фильтра:

- `kind` – тип фильтра, возможные варианты:
 - `access_levels` – поиск пропусков по заданным уровням доступа (`IdsValueSearchTermEntry`);
 - `accessible_areas` – поиск пропусков, обладающих правом доступа на указанные территории (`IdsSearchTermEntry`);
 - `accessible_readers` – поиск пропусков, обладающих правом прохода через указанные считыватели (`IdsSearchTermEntry`).

Фильтр для поиска пропусков по дополнительным полям персоны-владельца пропуска.

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByPersonAdditionalFieldsSearchTerm`.

Параметры фильтра:

- `additional_field[1..20]` – дополнительное поле пропуска для поиска (`StringValueSearchTermEntry`).

Фильтр для поиска пропусков по персональным данным персоны-владельца пропуска.

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByPersonAllInfoSearchTerm`.

Параметры фильтра:

- `gender` – фильтр по гендеру владельца:
 - `is_assigned` – задано ли значение (`AssignedSearchTermEntry`);
 - `gender` – значение гендера владельца:
 - `GENDER_UNSPECIFIED` – не указан;
 - `GENDER_MALE` – мужской;
 - `GENDER_FEMALE` – женский;
- `birth_date` – фильтр по дате рождения (`DateValueSearchTermEntry`);
- `birth_place` – фильтр по месту рождения (`StringValueSearchTermEntry`);
- `citizenship` – фильтр по гражданству (`IdsValueSearchTermEntry`);
- `address` – фильтр по адресу проживания (`StringValueSearchTermEntry`);
- `phone` – фильтр по номеру телефона (`StringValueSearchTermEntry`);
- `email` – фильтр по электронной почте (`StringValueSearchTermEntry`);
- `document_type` – фильтр по типу документа (`IdsValueSearchTermEntry`);
- `document_authority` – фильтр по по органу, выдавшему документ (`IdsValueSearchTermEntry`);

- document_serial – фильтр по серии документа (StringValueSearchTermEntry);
- document_number – фильтр по номеру документа (StringValueSearchTermEntry);
- document_issue_date – фильтр по дате выдачи документа (DateValueSearchTermEntry).

Фильтр для поиска пропусков по атрибутам привязанной к ним карте доступа.

type_url: type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByCardAttributesSearchTerm.

Параметры фильтра:

- card_code – код идентификации карты доступа. При поиске выполняется сопоставление кодов идентификации карт доступа в БД с указанным кодом. При сопоставлении учитывается системная настройка длины кода идентификации (сопоставление выполняется по настроенному в системе количеству байт из кода идентификации).

Фильтр для поиска пропусков по дополнительным полям пропусков.

type_url: type.googleapis.com/esprom.taurus.grpc.v1.persons.PassByPassAdditionalFieldsSearchTerm.

Параметры фильтра:

- descriptor_id – идентификатор дескриптора дополнительного поля;
- integer_value – фильтр по значению дополнительного поля целочисленного типа (Int64ValueSearchTermEntry);
- float_value – фильтр по значению дополнительного поля для чисел с плавающей точкой (DoubleValueSearchTermEntry);
- string_value – фильтр по значению дополнительного поля строчного типа (StringValueSearchTermEntry);
- boolean_value – фильтр по значению дополнительного поля булевого типа (BooleanValueSearchTermEntry);
- date_time_value – фильтр по значению дополнительного поля для даты и времени (TimestampValueSearchTermEntry);
- date_value – фильтр по значению дополнительного поля для даты (DateValueSearchTermEntry).

5.16. PersonLocationService

Файл person_location.proto предназначен для отслеживания персон на территории.

5.16.1. Метод GetPersonLocationCounters

Запрос на получение информации количестве персон на определенной территории. Запрос может быть выполнен только с использованием токена доступа.

```
{
  "control_area_id": 101
}
```

- control_area_id – идентификатор территории.

Ответ:

```
{
  "counters": [
    {
      "control_area_id": 101,
      "pass_category_id": 2,
      "people_count": 34
    },
    {
      "control_area_id": 101,
      "pass_category_id": 3,
      "people_count": 2
    },
    ...
  ]
}
```

- control_area_id – идентификатор территории отслеживания;
- pass_category_id – идентификатор категории пропусков, которая находится на территории;
- people_count – количество персон данной категории на указанной территории.

5.16.2. Метод GetPersonLocations

Запрос на получение информации о проходах категорий пропусков, на определенную территорию под определенным подразделением. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_category_id": {
    "value": 2
  },
  "control_area_id": {
    "value": 101
  },
  "organization_node_id": {
    "value": 2
  }
}
```

- pass_category_id – идентификатор категорий пропусков;
- control_area_id – территория, на которой проходили;
- organization_node_id – подразделение к которому принадлежит проходящая персона.

Ответ:

```
{
  "locations": [
    {
      "person_id": 126,
      "pass_id": 222,

```

```

    "control_area_id": 101,
    "pass_point_sdn": {
      "value": 367
    },
    "pass_time": {
      "seconds": 0,
      "nanos": 0
    },
    "message_type_id": {
      "value": 22
    }
  },
  {
    "person_id": 34,
    "pass_id": 123,
    "control_area_id": 101,
    "pass_point_sdn": {
      "value": 0
    },
    "pass_time": {
      "seconds": 0,
      "nanos": 0
    },
    "message_type_id": {
      "value": 22
    }
  },
  ...
]
}

```

- person_id – идентификатор проходящей персоны;
- pass_id – идентификатор используемого пропуска;
- control_area_id – идентификатор области контроля;
- pass_point_sdn – идентификатор точки прохода;
- pass_time – время прохода;
- message_type_id – идентификатор типа события о проходе.

5.16.3. Метод PersonLocationChanged

Запрос на подписку отслеживания прохода. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```

{
  "locations": [
    {
      "person_id": 126,
      "pass_id": 222,
      "control_area_id": 101,
      "pass_point_sdn": {
        "value": 367
      },
      "pass_time": {

```

```
    "seconds": 0,
    "nanos": 0
  },
  "message_type_id": {
    "value": 22
  }
},
{
  "person_id": 34,
  "pass_id": 123,
  "control_area_id": 101,
  "pass_point_sdn": {
    "value": 0
  },
  "pass_time": {
    "seconds": 0,
    "nanos": 0
  },
  "message_type_id": {
    "value": 22
  }
},
...
]
}
```

5.16.4. Метод PersonLocationCounterChanged

Запрос на подписку изменения количества персон на территориях. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "counters": [
    {
      "control_area_id": 101,
      "pass_category_id": 2,
      "people_count": 34
    },
    {
      "control_area_id": 101,
      "pass_category_id": 3,
      "people_count": 2
    },
    ...
  ]
}
```

5.17. SecurityControlGroupService

Файл security_control_group.proto предназначен для получения информации о групп управления охраной (ГУО).

Подробнее о ГУО можно прочитать в «Бастион-3. Руководство администратора».

5.17.1. Метод GetSecurityControlGroups

Запрос на получения полного списка ГУО. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "groups": [
    {
      "id": 1,
      "name": "Программная ГУО"
    },
    {
      "id": 2,
      "name": "Программная ГУО 2"
    },
    {
      "id": 3,
      "name": "Программная ГУО 3"
    },
    ...
  ]
}
```

- id – идентификатор ГУО;
- name – наименование ГУО.

5.17.2. Метод GetSecurityControlGroup

Запрос на получение информации о ГУО по идентификатору. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "security_control_group_id": 5
}
```

- id – идентификатор ГУО.

Пример ответа:

```
{
  "security_control_group":
  {
    "id": 5,
    "name": {
      "value": "Аппаратная ГУО"
    },
  },
}
```

```
    "hardware": {
      "din": 102,
      "address": 5,
      "can_modify": false
    }
  }
}
```

- id – идентификатор ГУО;
- name – наименование ГУО;
- din – идентификатор драйвера, к которой принадлежит ГУО;
- address – адрес ГУО;
- can_modify – признак возможности ручного редактирования состава группы.

Поле "hardware" указывается только, если ГУО является аппаратной.

5.17.3. Метод GetSecurityControlGroupContent

Запрос на получения состава ГУО по идентификатору. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "security_control_group_id": 5
}
```

- id – идентификатор ГУО.

Пример ответа, если ГУО является аппаратной:

```
{
  "hardware": {
    "entries": [
      {
        "sdn": 202,
        "can_disarm": false
      },
      {
        "sdn": 200,
        "can_disarm": false
      }
    ]
  }
}
```

- sdn – идентификатор устройства (раздела, группы разделов и т.п.), входящего в ГУО;
- can_disarm – Признак наличия прав на снятие с охраны.

Пример ответа, если ГУО является программной:

```
{
  "software": {
    "hardware_security_control_group_ids": [
      7,
      8
    ]
  }
}
```

- hardware_security_control_group_ids – содержит список идентификаторов аппаратных ГУО, входящих в состав программной ГУО.

5.18. StopListService

Файл stop_list.proto предназначен для управления стоп-листа.

5.18.1. Метод GetBlockedPersons

Запрос на получение персон, находящихся в стоп-листе. Запрос может быть выполнен только с использованием токена доступа.

Для получения определенного списка персон требуется использовать запрос с указанием идентификаторов персон:

```
{
  "by_person_ids": {
    "person_ids": [
      141,
      108
    ]
  }
}
```

В ответ войдут все персоны находящиеся в стоп-листе с указанными идентификаторами:

```
{
  "persons": [
    {
      "person_id": 107,
      "block_date": {
        "seconds": "1713109392",
        "nanos": 815363000
      },
      "reason": {
        "value": "в отпуске"
      }
    },
    {
      "person_id": 141,
      "block_date": {
        "seconds": "1713109400",
```

```

        "nanos": 74101000
      },
      "reason": {
        "value": "не указано"
      }
    }
  ]
}

```

- person_id – идентификатор персоны;
- block_date – дата добавления в стоп-лист;
- reason – причина добавления в стоп-лист.

Для получения полного списка персон, требуется отправить следующий запрос:

```

{
  "empty": {}
}

```

5.18.2. Метод GetBlockedPersonByPersonId

Запрос на получение информации о блокировке персоны с помощью идентификатора. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "person_id": 107
}

```

Ответ:

```

{
  "person": {
    "person_id": 107,
    "block_date": {
      "seconds": "1713109887",
      "nanos": 373867000
    },
    "reason": {
      "value": "в отпуске"
    }
  }
}

```

5.18.3. Метод AddPersonToStopList

Запрос на добавление персоны в стоп-лист. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_id": 107
  "reason": {
    "value": "длительный отпуск"
  }
}
```

- person_id – идентификатор персоны, которую нужно отправить в стоп-лист;
- reason – причина добавления в стоп-лист.

При успешной отправке персоны в стоп-лист, получим пустое сообщение.

5.18.4. Метод RemovePersonFromStopList

Запрос на удаление персоны из стоп-листа. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_id": 107
}
```

- person_id – идентификатор персоны, которую нужно удалить из стоп-листа.

При успешном удалении персоны из стоп-листа, получим пустое сообщение.

5.18.5. Метод BlockedPersonChanged

Запрос на подписку об изменениях в стоп-листе. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

При изменении в стоп-листе получим следующее сообщение:

```
{
  "person_id": 107,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- person_id – идентификатор персоны, которую добавили или удалили из стоп-листа;
- change_type – тип изменения стоп-листа (всегда будет иметь значение ENTITY_CHANGE_TYPE_UPDATE – обновлен).

5.19. TimeBlockService

Файл time_block.proto для получения информации о временных блоках.

5.19.1. Метод GetTimeBlock

Запрос на получение информации о временном блоке. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "time_block_id": 1
}
```

- `time_block_id` – идентификатор временного блока.

Ответное сообщение содержит всю информацию о временном блоке:

```
{
  "time_block": {
    "time_zones": [
      {
        "start_time": {
          "hours": 0,
          "minutes": 0,
          "seconds": 0,
          "nanos": 0
        },
        "end_time": {
          "hours": 23,
          "minutes": 59,
          "seconds": 59,
          "nanos": 0
        },
        "work_days_mask": 127,
        "holiday_mask": 3
      }
    ],
    "id": 11,
    "physical_number": 10,
    "is_weak": false,
    "name": "Скользкий блок",
    "consider_holidays": false,
    "period": 2,
    "start_date": {
      "year": 2024,
      "month": 4,
      "day": 14
    }
  }
}
```

- `time_zones` – содержит список временных зон:
 - `start_time` – время начала действия временной зоны:
 - `hours` – час;
 - `minutes` – минута;

- seconds – секунда;
- nanos – миллисекунда;
- end_time – время окончания действия временной зоны;
- work_days_mask – рабочие дни в рабочей неделе;
- holiday_mask – рабочие дни в выходные;
- id – идентификатор временного блока;
- physical_number – физический номер;
- is_weak – указатель на автоматическое создание;
- name – наименование временного блока (ограничение 100 символов);
- consider_holidays – учитывать ли праздничные дни;
- period – периодичность, относится ко скользящему временному блоку;
- start_date – дата с которой будет вестись расчет периодичности скользящего блока.

5.19.2. Метод GetTimeBlocks

Запрос на получение полного списка временных блоков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "time_blocks": [
    {
      "time_zones": [
        {
          "start_time": {
            "hours": 15,
            "minutes": 5,
            "seconds": 0,
            "nanos": 0
          },
          "end_time": {
            "hours": 23,
            "minutes": 59,
            "seconds": 59,
            "nanos": 0
          },
          "work_days_mask": 49,
          "holiday_mask": 1
        }
      ],
      "id": 1,
      "physical_number": 1,
      "is_weak": false,
      "name": "Круглосуточно (№1)",
      "consider_holidays": false,
      "period": 7,
      "start_date": null
    },
    {
      "time_zones": [
        {
          "start_time": {
```

```

        "hours": 0,
        "minutes": 0,
        "seconds": 0,
        "nanos": 0
    },
    "end_time": {
        "hours": 23,
        "minutes": 59,
        "seconds": 59,
        "nanos": 0
    },
    "work_days_mask": 71,
    "holiday_mask": 3
  }
],
"id": 3,
"physical_number": 2,
"is_weak": true,
"name": "Блок №2 (авто)",
"consider_holidays": true,
"period": 7,
"start_date": null
},
...
]
}

```

5.19.3. Метод TimeBlockChanged

Запрос на подписку об изменениях во временных блоках. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```

{
  "time_block_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}

```

- time_block_id – идентификатор временного блока, который изменился;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – временной блок создан;
 - ENTITY_CHANGE_TYPE_UPDATE – временной блок обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – временной блок удален.

5.20. AdditionalFieldService

Файл additional_field.proto предназначен для получения информации о дескрипторах дополнительных полей драйвера.

5.20.1. GetAdditionalFieldDescriptor

Запрос на получение информации о дескрипторе. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "descriptor_id": 1
}
```

- descriptor_id – идентификатор дескриптора;

Ответ:

```
{
  "field_descriptor": {
    "id": 1,
    "name": "деск",
    "description": {
      "value": "порт"
    }
  }
}
```

- id – идентификатор дескриптора;
- name – наименование дескриптора;
- description – назначение дескриптора.

5.20.2. GetAdditionalFieldDescriptors

Запрос на получение полного списка дескрипторов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ содержит список дескрипторов в системе:

```
{
  "descriptors": [
    {
      "id": 1,
      "name": "деск 1",
      "description": {
        "value": "порт"
      }
    },
    {
      "id": 2,
      "name": "деск 2",
      "description": {
        "value": "наличие доступа"
      }
    }
  ],
}
```

```

    ...
  ]
}

```

5.20.3. GetAdditionalFieldValues

Запрос для получения значения дескриптора. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса, если значения дескриптора нужно только от одного устройства:

```

{
  "by_device": {
    "sdn": 101
  }
}

```

- sdn – идентификатор устройства.

Если необходимо значения дескрипторов от нескольких устройств, используется следующий запрос, где необходимо перечислить идентификаторы устройств:

```

{
  "by_devices": {
    "sdns": [
      101,
      365
    ]
  }
}

```

Так же можно получить значения дескрипторов для всех устройств экземпляра драйвера:

```

{
  "by_driver": {
    "din": 100
  }
}

```

- din – идентификатор драйвера.

Результат любого из запросов будет список значений дескрипторов:

```

{
  "values": [
    {
      "sdn": 365,
      "descriptor_id": 1,
      "boolean": true
    },

```

```
{
  {
    "sdn": 365,
    "descriptor_id": 2,
    "integer": "5432"
  }
]
```

- sdn – идентификатор устройства.
- descriptor_id – идентификатор дескриптора;
- Следующее поле может отличаться в зависимости от того, какой тип значений указан в дескрипторе:
 - boolean – булево значение;
 - integer – целое числовое;
 - float – дробное число;
 - string – текст;
 - date_time – дата и время;
 - date – дата;
 - time_span – время.

5.20.4. GetAdditionalFieldValueChanged

Запрос на подписку об изменениях дескрипторах. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "value": {
    "sdn": 345,
    "descriptor_id": 101,
    "boolean": false,
  },
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- sdn – идентификатор устройства связанного с дескриптором;
- descriptor_id – идентификатор дескриптора, который изменился;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – дескриптор создан;
 - ENTITY_CHANGE_TYPE_UPDATE – дескриптор обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – дескриптор удален.

5.21. DriverInfoService

Файл `driver_info.proto` предназначен для получения информации о драйверах и устройствах.

5.21.1. Глоссарий

5.21.1.1. Тип драйвера

Класс `esprom.taurus.grpc.v1.drivers.DriverType`.

Соответствует отдельной интегрируемой в Бастион-3 системе, например: Elsys, C2000, Заря, Macroscop и т.п. Получения списка установленных/поддерживаемых системой типов драйверов доступно через метод `GetDriverTypes`.

Основные поля:

- `driver_id` – уникальный идентификатор, константа, задаётся при начале разработки интеграции со сторонней системой;
- `name` – название типа драйвера;
- `version` – версия типа драйвера.

5.21.1.2. Базовый тип устройства

Сейчас отсутствует в gRPC, по сути является статичным справочником. Описывает абстрактный тип устройства, например: дверь, турникет, контроллер, считыватель, камера и т.п. Список доступных типов можно просмотреть в БД в таблице `driver.device_types`;

5.21.1.3. Сервер (хост) оборудования

Класс `esprom.taurus.grpc.v1.drivers.DriverHost`.

Информация об отдельном компьютере/сервере в системе, который может выполнять роль серверов оборудования для конкретного типа драйверов. Получение списка доступно через метод `GetDriverHosts`. Список серверов оборудования всегда содержит хост, на котором работает сервер системы.

Основные поля:

- `socket_id` – уникальный идентификатор сервера оборудования, генерируется при добавлении сервера оборудования в систему;
- `name` – сетевое имя или IP-адрес сервера оборудования.

5.21.1.4. Экземпляр (инстанс) драйвера

Класс `esprom.taurus.grpc.v1.drivers.DriverInstance`.

Экземпляр конкретного типа драйвера; в зависимости от типа драйвера в систему может быть добавлен один, один на сервер оборудования и несколько экземпляров (для масштабирования) драйвера. Получение списка доступно через метод `GetDriverInstances`.

Основные поля:

- `dip` – уникальный идентификатор экземпляра драйвера, генерируется при добавлении экземпляра в систему;

- `socket_id` – идентификатор сервера оборудования, к которому относится (на котором должен работать) экземпляр драйвера; Если поле не задано, то экземпляр всегда относится к хосту - серверу системы (в основном это экземпляры системных типов драйверов);
- `driver_id` – идентификатор типа драйвера, за интеграция устройств которого отвечает данный экземпляр драйвера;
- `name` – имя экземпляра драйвера.

5.21.1.5. Устройство

Класс `esprom.taurus.grpc.v1.drivers.Device`.

Устройство интегрируемое в систему рамках конкретного экземпляра драйвера, может являться непосредственным источником событий или выполнять команды. Получение списка доступно через метод `GetDriverInstanceDevices`, либо через сервис поиска устройств (`esprom.taurus.grpc.v1.drivers.SearchDeviceService`).

Основные поля:

- `sdn` – уникальный идентификатор устройства, генерируется при добавлении устройства в систему;
- `parent_sdn` – идентификатор родительского устройства, используется для построения иерархии устройств. В рамках одного экземпляра драйвера всегда есть только одно корневое устройство – логическое устройство экземпляра драйвера, для которого не задан `parent_sdn`, для остальных устройств экземпляра драйвера это поле обязательно имеет значение;
- `din` – идентификатор экземпляра драйвера, к которому относится устройство;
- `name` – имя устройства.

5.21.1.6. Тип события устройства

Класс `esprom.taurus.grpc.v1.drivers.MessageType`.

Описание конкретного типа событий, который может генерироваться устройствами заданного типа драйвера с заданным базовым типом. Получение списка типов событий, относящихся к конкретному типу драйверов, доступно через метод `GetDriverMessageTypes`, либо через анализ ddt-файлов.

Основные поля:

- `driver_id` – идентификатор типа драйвера, к которому относится устройство, которое может генерировать данный тип события (далее устройство-источник)
- `device_type` – код базового типа устройства, к которому относится устройство-источник;
- `code` – уникальный в рамках типа драйвера и базового типа устройства код для типа событий;
- `text` – шаблон текста сообщения о событии;
- `message_profile_id` – идентификатор профиля событий (ссылка на `MessageProfile`).

5.21.1.7. Тип команды_действия устройства

Класс `esprom.taurus.grpc.v1.drivers.ActionType`.

Описание конкретного типа команды, которое поддерживается устройствами заданных базового типа и типа драйвера. Получение списка команд, относящихся к конкретному типу драйверов, доступно через метод `GetDriverActionTypes`, либо через анализ ddt-файлов.

Основные поля:

- `driver_id` – идентификатор типа драйвера, к которому относится устройство, которое может выполнять данный тип команды (далее целевое устройство)
- `device_type` – код базового типа устройства, к которому относится целевое устройство;
- `code` – уникальный в рамках типа драйвера и базового типа устройства код для типа событий;
- `name` – имя/название команды.

5.21.1.8. Тип группы устройств

Класс `esprom.taurus.grpc.v1.drivers.DeviceGroupType`.

Описание конкретного типа группы устройств. Получение списка всех типов групп устройств, или только относящихся к конкретному типу драйверов, доступно через метод `GetDeviceGroupTypes`.

Основные поля:

- `code` – уникальный в рамках типа драйвера код для типа группы устройств;
- `driver_id` – идентификатор типа драйвера, к которому относится тип группы устройств;
- `name` – имя типа группы устройств.

5.21.2. Метод `GetDriverHosts`

Запрос на получение списка серверов оборудования. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "driver_hosts": [
    {
      "socket_id": 100,
      "name": "Server"
    },
    {
      "socket_id": 110,
      "name": "DESKTOPLinux"
    },
    ...
  ]
}
```

- `socket_id` – идентификатор сервера оборудования;
- `name` – наименование сервера.

5.21.3. Метод `GetDriverTypes`

Запрос получение списка о типах драйверов. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "include_system_types": false
}
```

- `include_system_types` – стоит ли включать в список типы системных драйверов.

Пример ответа:

```
{
  "driver_types": [
    {
      "driver_id": 12,
      "name": "Бастион-3 - Face",
      "version": {
        "value": "2.0.1"
      }
    },
    {
      "driver_id": 33,
      "name": "Бастион-3 - Алкорамка",
      "version": {
        "value": "2024.1"
      }
    },
    {
      "driver_id": 43,
      "name": "Бастион-3 - Сириус",
      "version": {
        "value": "1.0.1"
      }
    }
  ]
}
```

- `driver_id` – идентификатор драйвера;
- `name` – наименование драйвера;
- `version` – версия драйвера.

5.21.4. Метод `GetDriverInstances`

Запрос для получения экземпляра драйверов. Запрос может быть выполнен только с использованием токена доступа.

Для получения списка экземпляров типов драйвера с указанного сервера оборудования, нужно использовать запрос с указанием идентификатора сервера оборудования:

```
{
  "socket_id": {
    "value": 100
  }
}
```

Для получения экземпляров драйверов определенного типа, нужно использовать запрос с указанием идентификатора драйвера:

```
{
  "driver_id": {
    "value": 88888
  }
}
```

Эти запросы можно объединять, таким образом получить экземпляры драйверов определенного типа на указанном сервере оборудования:

```
{
  "driver_id": {
    "value": 88888
  },
  "socket_id": {
    "value": 100
  }
}
```

Результатом всегда будет список типов экземпляров драйвера:

```
{
  "driver_instances": [
    {
      "din": 100,
      "socket_id": {
        "value": 100
      },
      "driver_id": 12,
      "name": "Face"
    },
    {
      "din": 101,
      "socket_id": {
        "value": 100
      },
      "driver_id": 33,
      "name": "Алкорамка"
    },
    {
      "din": 102,
      "socket_id": {
        "value": 100
      },
      "driver_id": 43,
      "name": "Сириус"
    }
  ]
}
```

- `din` – идентификатор экземпляра драйвера;
- `socket_id` – идентификатор сервера оборудования;
- `driver_id` – идентификатор типа драйвера;
- `name` – наименование экземпляра драйвера.

5.21.5. Метод `GetDriverInstanceDevice`

Запрос на получение устройств экземпляра драйвера. Запрос может быть выполнен только с использованием токена доступа. Указание идентификатора экземпляра драйвера является обязательным для всех запросов.

Если нужно найти все устройства определенного типа, то нужно использовать следующий запрос:

```
{
  "by_device_type": {
    "value": 101
  },
  "din": 100
}
```

- `din` – идентификатор экземпляра драйвера;
- `by_device_type` – поиск с использованием идентификатора типа драйвера.

Поиск можно осуществить с использованием идентификаторов устройств:

```
{
  "by_device_sdns": {
    "ids": [
      345,
      101,
      23,
      34
    ]
  },
  "din": 100
}
```

- `by_device_sdns` – поиск с использованием идентификаторов устройств.

Если необходимо найти устройства различных типов драйвера, то нужно указать список типов драйверов для поиска:

```
{
  "by_device_types": {
    "ids": [
      22,
      33
    ]
  },
  "din": 100
}
```

```
}

```

- `by_device_types` – поиск с использованием идентификаторов типов драйверов.

Результатом будет набор сообщений содержащий в себе информацию об устройстве:

```
{
  "device": {
    "sdn": 376,
    "din": 100,
    "parent_sdn": {
      "value": 201
    },
    "name": "Сервер внешней системы 4",
    "type": 26,
    "address": 0,
    "is_active": true,
    "time_zone_code": 0
  }
}
```

- `sdn` – идентификатор устройства;
- `din` – идентификатор экземпляра драйвера;
- `parent_sdn` – идентификатор родительского устройства;
- `name` – наименование устройства;
- `type` – тип устройства;
- `address` – адрес устройства;
- `is_active` – активно ли устройство;
- `time_zone_code` – код временной зоны, где находится устройство.

5.21.6. Метод `GetDevice`

Запрос на получение информации об устройстве. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "sdn": 364
}
```

- `sdn` – идентификатор устройства.

Ответ содержит информацию об устройстве:

```
{
  "device": {
    "sdn": 364,
```

```

        "din": 100,
        "parent_sdn": {
            "value": 201
        },
        "name": "Виртуальная точка прохода 1",
        "type": 3,
        "address": 0,
        "is_active": true,
        "time_zone_code": 0
    }
}

```

5.21.7. Метод `GetDriverActionTypes`

Запрос на получение всех типов действия драйверов. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "driver_id": 22
}

```

- `driver_id` – идентификатор типа драйвера.

Вами будут получены сообщения о возможных действиях данного типа драйвера:

```

{
  "action_type": {
    "driver_id": 22,
    "device_type": 3,
    "code": 2,
    "name": "Нормальный режим"
  }
}

```

- `driver_id` – идентификатор типа драйвера.
- `device_type` – тип устройства;
- `code` – код действия;
- `name` – наименование действия.

5.21.8. Метод `GetDriverMessageTypes`

Запрос на получение о возможных событиях от экземпляров типа драйвера. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "driver_id": 22
}

```

```
}

```

- driver_id – идентификатор типа драйвера.

Ответные сообщения содержат информацию о возможных событиях устройства указанного типа драйвера:

```
{
  "message_type": {
    "driver_id": 22,
    "device_type": 3,
    "code": 3,
    "text": "Штатный выход %s1",
    "message_profile_id": 5,
    "pass_direction": "PASS_DIRECTION_TYPE_UNSPECIFIED",
    "credential_check_result": "CREDENTIAL_CHECK_RESULT_UNSPECIFIED",
    "credential_check_result_group": null
  }
}
```

- driver_id – идентификатор типа драйвера;
- device_type – тип устройства;
- code – код типа события;
- text – информационное сообщение;
- message_profile_id – идентификатор профиля сообщения;
- pass_direction – направление прохода:
 - PASS_DIRECTION_TYPE_UNSPECIFIED – не указано;
 - PASS_DIRECTION_TYPE_IN – вход;
 - PASS_DIRECTION_TYPE_OUT – выход.
- credential_check_result – результат проверки полномочий на доступ пользователя СКУД через точку прохода:
 - CREDENTIAL_CHECK_RESULT_UNSPECIFIED – не определен;
 - CREDENTIAL_CHECK_RESULT_ACCESS_VERIFIED – пропуск имеет полномочия на доступ;
 - CREDENTIAL_CHECK_RESULT_ACCESS_DENIED – пропуск не имеет полномочия на доступ;
 - CREDENTIAL_CHECK_RESULT_UNKNOWN_CREDENTIAL – карта неизвестна;
 - CREDENTIAL_CHECK_RESULT_BLOCKED_PASS – пропуск заблокирован;
 - CREDENTIAL_CHECK_RESULT_EXPIRED_PASS – пропуск просрочен;
 - CREDENTIAL_CHECK_RESULT_TIME_ZONE_VIOLATION – пропуск не имеет полномочия на доступ из-за нарушения временной зоны;
 - CREDENTIAL_CHECK_RESULT_PENDING_INITIALIZATION – пропуск имеет полномочие, но не был доставлен в контроллер.

- `credential_check_result_group` – строковый ключ для группы результатов проверки полномочий.



Внимание!

Для определения успешного прохода необходимо смотреть на поле `credential_check_result`, оно должно иметь значение `CREDENTIAL_CHECK_RESULT_ACCESS_VERIFIED`. Проход считается не успешным если значение `credential_check_result` не равно значению `CREDENTIAL_CHECK_RESULT_ACCESS_VERIFIED` и не равно значению `CREDENTIAL_CHECK_RESULT_UNSPECIFIED`. При этом проход возможен только если значение поля `pass_direction` не равно `PASS_DIRECTION_TYPE_UNSPECIFIED`.

5.21.9. Метод `ExecuteDeviceCommand`

Запрос на выполнение действия. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "action_type_code": 13978182,
  "parameters": {
    "value": ""
  },
  "sdn": 354
}
```

- `action_type_code` – код типа события;
- `parameters` – строковое значение для передачи параметров события;
- `sdn` – идентификатор устройства, на котором произошло событие.

При успешном выполнении придет пустое сообщение.

5.21.10. Метод `GetDeviceState`

Запрос на получение состояния устройства. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "sdn": 354
}
```

- `sdn` – идентификатор устройства, состояние которого запрашивается.

Пример ответа:

```
{
```

```
"state": {
  "simple": {
    "id": 0,
    "code": 0,
    "time": {
      "seconds": "1713131098",
      "nanos": 524790300
    }
  }
}
```

- id – идентификатор состояния;
- code – код состояния;
- time – время наступления текущего состояния.

Полный перечень возможных состояний устройств приведен в [приложении](#).

5.21.11. Метод ConfigurationChanged

Запрос на отслеживание изменений конфигураций устройств. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответного сообщения:

```
{
  "Changes": {
    "889222": {
      "Changes": {
        "100": {
          "new_devices": [
            377,
            378,
            379,
            380
          ],
          "updated_device": [],
          "deleted_device": [],
          "din": 100
        }
      },
      "driver_id": 889222
    }
  },
  "driver_instances_changed": false
}
```

- Changes – содержит информацию какие драйвера затронули изменения:
 - Changes – содержит информацию об изменениях экземпляров драйвера:
 - new_device – содержит идентификаторы новых устройств драйвера;
 - updated_device – содержит идентификаторы обновленных устройств;
 - deleted_device – содержит информацию об удаленных устройствах;

- `din` – идентификатор типа экземпляра драйвера;
- `driver_id` – идентификатор экземпляра драйвера;
- `driver_instances_changed` – был ли изменен экземпляр драйвера.

5.21.12. Метод `GetDeviceGroupTypes`

Запрос на получение типов групп устройств. Запрос может быть выполнен только с использованием токена доступа.

Для получения списка всех типов групп устройств нужно использовать пустое тело запроса. Для получения списка типов групп устройств для экземпляров драйверов определенного типа, нужно использовать запрос с указанием идентификатора драйвера:

```
{
  "driver_id": {
    "value": 268
  }
}
```

В ответ будут получены сообщения с информацией о типах групп устройств:

```
{
  "code": 0,
  "driver_id": 268,
  "name": {
    "value": "Группа для приборов"
  }
}
```

- `code` – код типа группы, уникальный в рамках типа драйвера;
- `driver_id` – идентификатор типа драйвера;
- `name` – имя типа группы.

5.21.13. Метод `GetDeviceGroupMembers`

Запрос на получение содержимого группы устройств. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "type_code": 5,
  "owner_sdn": 1779
}
```

- `type_code` – код типа группы устройств;
- `owner_sdn` – идентификатор устройства-владельца группы.

Ответ:

```
{
  "device_group_members": [
    1785,
    1786,
    1784
  ]
}
```

- device_group_members – перечисление идентификаторов устройств, входящих в группу.

5.21.14. Метод GetDeviceGroupOwners

Запрос на получение всех владельцев групп устройств. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "type_code": 7,
  "member_sdn": 1785
}
```

- type_code – код типа группы устройств;
- member_sdn – идентификатор устройства-участника группы.

Ответ:

```
{
  "device_group_owners": [
    1832,
    1833
  ]
}
```

- device_group_owners – перечисление идентификаторов всех устройств-владельцев групп устройств.

5.21.15. Метод GetReaderToPassPointLinks

Запрос на получение списка привязок считывателей к точкам прохода.

Пример запроса:

```
{
  "driver_id": 356
}
```

- driver_id - возможная фильтрация списка привязок по типу драйвера, к которому принадлежат считыватели. Если не указано то возвращается список по всем типам драйверов.

Результатом будет набор сообщений, содержащий в себе информацию о привязке считывателя к точке прохода:

```
{
  "reader_sdn": 348168,
  "pass_point_sdn": 348167,
  "role": "READER_ROLE_INPUT"
}
```

- reader_sdn - идентификатор считывателя, привязанного к точке прохода;
- pass_point_sdn - идентификатор точки прохода, к которой привязан считыватель;
- role - роль считывателя, где "READER_ROLE_INPUT" - входной считыватель, "READER_ROLE_OUTPUT" - выходной считыватель, "READER_ROLE_UNSPECIFIED" - не указано.

5.21.16. Метод GetMessageType

Запрос на получение информации о типе события. Запрос может быть выполнен только с использованием токена доступа.

Если нужно найти тип события по его идентификатору, то нужно использовать следующий запрос:

```
{
  "by_message_type_id": {
    "message_type_id": 104
  }
}
```

- message_type_id – идентификатор типа события.

Поиск можно также осуществить с использованием составного ключа типа события:

```
{
  "by_message_type_key": {
    "driver_id": 1,
    "device_type": 0,
    "code": 4
  }
}
```

- driver_id – идентификатор типа драйвера;
- device_type – идентификатор типа устройства;
- code – код типа сообщения.

Ответ содержит информацию об типе события:

```
{
  "message_type": {
    "driver_id": 1,
    "device_type": 0,
    "code": 4,
    "text": "Изменение общих настроек пропускного режима",
    "message_profile_id": 4,
    "pass_direction": "PASS_DIRECTION_TYPE_UNSPECIFIED",
    "credential_check_result": "CREDENTIAL_CHECK_RESULT_UNSPECIFIED",
    "credential_check_result_group": null,
    "message_type_id": 104
  }
}
```

- driver_id – идентификатор типа драйвера;
- device_type – идентификатор типа устройства;
- code – код типа сообщения;
- text – шаблон текста сообщения;
- message_profile_id – идентификатор профиля сообщения;
- pass_direction – направление прохода:
 - PASS_DIRECTION_TYPE_UNSPECIFIED – направление не задано;
 - PASS_DIRECTION_TYPE_IN – вход в помещение;
 - PASS_DIRECTION_TYPE_OUT – выход из помещения;
- credential_check_result – результат проверки полномочий на доступ пользователя СКУД через точку прохода:
 - CREDENTIAL_CHECK_RESULT_UNSPECIFIED – результат проверки не определён;
 - CREDENTIAL_CHECK_RESULT_ACCESS_VERIFIED – пропуск имеет полномочие на доступ;
 - CREDENTIAL_CHECK_RESULT_ACCESS_DENIED – пропуск не имеет полномочия на доступ;
 - CREDENTIAL_CHECK_RESULT_UNKNOWN_CREDENTIAL – карта неизвестна;
 - CREDENTIAL_CHECK_RESULT_BLOCKED_PASS – пропуск заблокирован;
 - CREDENTIAL_CHECK_RESULT_EXPIRED_PASS – пропуск просрочен;
 - CREDENTIAL_CHECK_RESULT_TIME_ZONE_VIOLATION – пропуск не имеет полномочия на доступ из-за нарушения временной зоны;
 - CREDENTIAL_CHECK_RESULT_PENDING_INITIALIZATION – пропуск имеет полномочие, но не был доставлен в контроллер;
- credential_check_result_group – дополнительная информация о группе типов событий для проверки полномочия пользователей СКУД;
- message_type_id – идентификатор типа события.

5.22. SearchDevicesService

Файл search_device.proto предназначен для поиска устройств сервера оборудования.

5.22.1. SearchDevices

Запрос на поиск устройств с определенными критериями. Запрос может быть выполнен только с использованием токена доступа.

При поиске устройств в параметрах запроса (`SearchDevicesRequest`) можно указать дополнительные условия поиска (в поле `terms`), при этом сами условия поиска запаковываются в тип `Any` (см. [Приложение 1. Тип google.protobuf.Any](#)).

Ответное сообщение содержит информацию об устройствах, которые подошли под указанные фильтры:

```
{
  "device": {
    "sdn": 0,
    "din": 0,
    "parent_sdn": {
      "value": 0
    },
    "name": "",
    "type": 0,
    "address": 0,
    "is_active": false,
    "time_zone_code": 0
  }
}
```

5.22.2. Фильтры устройств

Фильтры устройств позволяют ограничить результаты поиска по основным параметрам. Условия поиска задаются с использованием стандартных элементов фильтрации, подробное описание которых приведено в [Приложении 3. Основные элементы фильтрации](#).

Поиск устройств по их основным атрибутам.

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.drivers.DeviceByMainPropertiesSearchTerm`.

Параметры фильтра:

- `device_types` – фильтр по типам устройств (`IdsSearchTermEntry`);
- `driver` – фильтр по драйверу устройства:
 - `by_types` – фильтр по типам драйверов устройства (`IdsSearchTermEntry`);
 - `by_instances` – фильтр по экземплярам драйверов устройства (`IdsSearchTermEntry`).

Поиск устройств по вхождению заданных подстрок в имя устройства.

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.drivers.DeviceByCommonPropertiesContainWordsSearchTerm`.

Параметры фильтра:

- `words` – список текстовых значений, которые будут проверяться на вхождение в имя устройств (`repeated string`).

Поиск устройств по их дополнительным полям.

`type_url`: `type.googleapis.com/esprom.taurus.grpc.v1.drivers.DeviceByAdditionalFieldSearchTerm`.

Параметры фильтра:

- `descriptor_id` – идентификатор дескриптора дополнительного поля;
- `integer_value` – фильтр по значению дополнительного поля целочисленного типа (`Int64ValueSearchTermEntry`);
- `float_value` – фильтр по значению дополнительного поля для чисел с плавающей точкой (`DoubleValueSearchTermEntry`);
- `string_value` – фильтр по значению дополнительного поля строчного типа (`StringValueSearchTermEntry`);
- `boolean_value` – фильтр по значению дополнительного поля булевого типа (`BooleanValueSearchTermEntry`);
- `date_time_value` – фильтр по значению дополнительного поля для даты и времени (`TimestampValueSearchTermEntry`);
- `date_value` – фильтр по значению дополнительного поля для даты (`DateValueSearchTermEntry`).

5.23. MessageInfoService

Файл `message_info.proto` предназначен для получения информации о сообщениях сервера системы.

5.23.1. Глоссарий1

5.23.1.1. Профиль события

Класс `esprom.taurus.grpc.v1.messages.MessageProfile`.

Описывает общие характеристики того или иного класса событий в системе. Получение списка всех профилей в системе доступно через метод `GetMessageProfiles`.

Основные поля:

- `id` – уникальный идентификатор профиля; для системных профилей является предустановленной константой, для пользовательских профилей генерируется при добавлении в систему.
- `priority` – приоритет событий, относящихся к этому профилю;
- `kind` – вид событий, относящихся к профилю, перечисление, основные значения: 1 – штатное событие, 2 – тревога, 3 – неисправность;
- `name` – имя профиля событий;
- `is_system` – признак системного предустановленного профиля.

5.23.2. Метод GetMessageProfile

Запрос на получение информации о профиле сообщения. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "profile_id": 35
}
```

```
}
```

- profile_id – идентификатор профиля сообщений.

Ответ:

```
{
  "profile": {
    "id": 35,
    "priority": 35,
    "kind": "MESSAGE_KIND_FAULT",
    "name": "Нарушение связи",
    "is_system": true
  }
}
```

- id – идентификатор профиля сообщений;
- priority – приоритет сообщения;
- kind – тип сообщения;
- name – сообщение;
- is_system – является ли сообщение системным.

5.23.3. Метод GetMessageProfiles

Запрос на получение списка профилей сообщения. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ содержит список всех профилей сообщения:

```
{
  "profiles": [
    {
      "id": 1,
      "priority": 1,
      "kind": "MESSAGE_KIND_NORMAL",
      "name": "Не показывать",
      "is_system": true
    },
    {
      "id": 35,
      "priority": 35,
      "kind": "MESSAGE_KIND_FAULT",
      "name": "Нарушение связи",
      "is_system": true
    },
    ...
  ]
}
```

5.23.4. Метод `GetUnconfirmedMessages`

Запрос на получение списка неподтверждённых сообщений. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответного сообщения:

```
{
  "message": {
    "id": 0,
    "session_id": 0,
    "server_session_id": 0,
    "event": {
      "sdn": 0,
      "message_code": 0,
      "time": {
        "seconds": 0,
        "nanos": 0
      },
      "card_code": 0,
      "detected_string": {
        "value": ""
      },
      "ext_int": 0,
      "ext_double": 0,
      "ext_string_1": {
        "value": ""
      },
      "ext_string_2": {
        "value": ""
      },
      "attached_image": {
        "value": []
      }
    },
    "profile_id": 0,
    "can_be_confirmed": false,
    "linked_cameras": [
      0
    ],
    "pass_direction": "PASS_DIRECTION_TYPE_UNSPECIFIED",
    "can_be_passage": false,
    "base_text": "",
    "prepared_text": "",
    "is_actual": false,
    "details": [
      {
        "type_url": "",
        "value": []
      }
    ]
  }
}
```

- `id` – идентификатор сообщения;
- `session_id` – идентификатор сессии;
- `server_session_id` – идентификатор сессии сервера;

- event – пришедшее событие:
 - sdn – идентификатор устройства, от которого было получено событие;
 - message_code – код события;
 - time – время события;
 - card_code – код карты, полученный в событии;
 - detected_string – распознанная строка. Константа подстановки %nb;
 - ext_int – дополнительное целое знаковой 4-х байтовое число. Используется при вычислении значений констант подстановки: %tv, %up, %ra, %rb, %av, %dl, %sc и %rr;
 - ext_double – дополнительное число двойной точности с плавающей запятой. Константа подстановки %ef;
 - ext_string_1 – первая дополнительная строка. Константа подстановки %s1;
 - ext_string_2 – вторая дополнительная строка. Константа подстановки %s2;
 - attached_image – привязанное к событию изображение;

5.23.5. Метод ConfirmMessages

Запрос на подтверждение оператором списка тревожных событий. Запрос может быть выполнен только с использованием токена доступа.

Для подтверждения тревожных событий требуется отправить список идентификаторов всех событий, которые требуется подтвердить:

```
{
  "confirm_parameters": [
    {
      "message_session_id": 101,
      "message_id": 345,
      "message_server_session_id": 7293ba11-1b76-4418-8798-c89ba12d9725,
      "sdn": 157
    },
    {
      "message_session_id": 101,
      "message_server_session_id": 7293ba11-1b76-4418-8798-c89ba12d9725,
      "message_id": 346,
      "sdn": 156
    },
    ...
  ]
}
```

- message_session_id – идентификатор сессии, где произошло событие;
- message_server_session_id – идентификатор сессии сервера оборудования, где произошло событие;
- message_id – идентификатор тревожного сообщения;
- sdn – идентификатор устройства, от которого произошло событие.

В качестве ответа придет сообщение с пустым телом.

5.23.6. Метод MessageProfileChanged

Запрос на получение уведомлений об изменении списка профилей событий. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

5.23.7. Метод NewMessageProcessed

Запрос на получение уведомлений об обработке системой списка новых сообщений. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа содержит список значений новых сообщений:

```
{
  "messages": [
    {
      "id": 0,
      "session_id": 0,
      "server_session_id": 0,
      "event": {
        "sdn": 0,
        "message_code": 0,
        "time": {
          "seconds": 0,
          "nanos": 0
        },
        "card_code": 0,
        "detected_string": {
          "value": ""
        },
        "ext_int": 0,
        "ext_double": 0,
        "ext_string_1": {
          "value": ""
        },
        "ext_string_2": {
          "value": ""
        },
        "attached_image": {
          "value": []
        }
      },
      "profile_id": 0,
      "can_be_confirmed": false,
      "linked_cameras": [
        0
      ],
      "pass_direction": "PASS_DIRECTION_TYPE_UNSPECIFIED",
      "can_be_passage": false,
      "base_text": "",
      "prepared_text": "",
      "is_actual": false,
      "details": [
        {
          "type_url": "",
          "value": []
        }
      ]
    }
  ]
}
```

```
]
}
```

5.23.8. Метод MessagesConfirmed

Запрос на получение уведомлений о подтверждении тревожных сообщений. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа со списком параметров тревожных сообщений:

```
{
  "confirm_parameters": [
    {
      "sdn": 0,
      "message_id": 0,
      "message_session_id": 0,
      "message_server_session_id": 0
    }
  ]
}
```

- sdn – идентификатор устройство, от которого поступило тревожное сообщение;
- message_id – идентификатор сообщения;
- message_session_id – идентификатор сессии, когда поступило тревожное сообщение;
- message_server_session_id – идентификатор сессии сервера, когда поступило тревожное сообщение.

5.23.9. Метод GenerateDeviceMessages

Запрос на генерацию события от устройства. Запрос может быть выполнен только с использованием токена доступа и при наличии модуля Messages версии 2025.5 и выше.

Для генерации событий необходимо наличие полномочия оператора: "Право на генерацию событий от устройств".

Пример запроса генерации события от устройства:

```
{
  "device_event": {
    "sdn": 204,
    "message_code": 1,
    "time": {
      "seconds": 1736934600,
      "nanos": 0
    },
    "cardCode": "1",
    "detected_string": {
      "value": "Штатный вход %nm %n1 %s1"
    },
    "ext_int": 42,
    "ext_double": 3.14159,
    "ext_string_1": {
```

```

        "value": "Иванов"
    },
    "ext_string_2": {
        "value": "Иван"
    },
    "attached_image": {
        "value":
"iVBORw0KGgoAAAANSUgUgAAAAEAAAABCAYAAAAfFcSJAAAADULEQVR42mNkYPhfDwAChwGA60e6kgAAAABJRU5ErkJg
gg=="
    }
}
}
}

```

5.24. ProtocolInfoService

Файл `protocol_info.proto` предназначен для получения информации о данных из протокола событий системы.

5.24.1. Метод GetMessages

Запрос на получение данных из протокола событий. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "terms": [
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByTimeSearchTerm",
      "value": "ChQKCAjQytiKBhAAEggI0KGajQYQAA=="
    },
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByDeviceInstancesSearchTerm",
      "value": "CggKBtMhmx6tEw=="
    }
  ],
  "detail_codes": [
    "AttachedPass", "AttachedCard", "AttachedCameras"
  ],
  "query_description": {
    "limit": {
      "value": 5
    },
    "offset": {
      "value": 4
    },
    "sort_descriptions": [
      {
        "field_name": "ProtocolMessage.gid",
        "sort_type": "SORT_TYPE_ASCENDING"
      }
    ]
  },
},

```

```

    "include_removed": false
  }

```

- `terms` – список запакованных в тип `google.protobuf.Any` (см. [Приложение 1](#)) фильтров, применяемых к событиям протокола. В списке обязательно должен быть указан как минимум один фильтр. Описание поддерживаемых фильтров приведено в разделе ["Фильтры событий"](#).
- `detail_codes` – набор строк-ключей для включения в результаты запроса дополнительной информации, связанной с событиями. Описание поддерживаемой дополнительной информации приведено в разделе ["Дополнительная информация"](#);
- `query_description` – дополнительные параметры запроса:
 - `limit` – ограничение на количество возвращаемых событий;
 - `offset` – значение сдвига, начиная с которого требуется получить события;
 - `sort_descriptions` – описание сортировки данных:
 - `field_name` – имя поля, по которому необходимо выполнить сортировку. Сортировка доступна по полям `"ProtocolMessage.gid"` и `"ProtocolMessage.time"`;
 - `sort_type` – необходимый порядок сортировки:
 - `SORT_TYPE_UNSPECIFIED` – не указан;
 - `SORT_TYPE_ASCENDING` – по возрастанию;
 - `SORT_TYPE_DESCENDING` – по убыванию;
 - `include_removed` – учитывать ли в результате запроса информацию об удалённых объектах.

Результатом будут сообщения с информацией о событиях из протокола, которые удовлетворяют условиям фильтрации, заданным в параметре `terms`.

Пример:

```

{
  "message": {
    "details": [
      {
        "key": "AttachedCard",
        "value": {
          "type_url": "type.googleapis.com/esprom.taurus.grpc.v1.persons.Card",
          "value": "CKnKAXD/8qqBwA4aBwj/8qqBwA4iDAiTmfOEBhCgloiLATAEOAE="
        }
      },
      {
        "key": "AttachedPass",
        "value": {
          "type_url": "type.googleapis.com/esprom.taurus.grpc.v1.persons.Pass",
          "value":
"CN7QBxDDugQYZyAEKgQIqcoDMgIIDjgBQgwImJjzhAYQ8ImbgwJKDAiN8PeEBhCw0NLRALIMCJOZ84QGELCqv6cBegMI
pH6CAQsI1NeBjQYQ+JSraA=="
        }
      }
    ],
    "gid": 33720117,
    "id": 422972,
    "session_id": 6,
    "server_session_id": 4654,
    "time": {
      "seconds": "1633035790",

```

```

        "nanos": 0
    },
    "source_device_id": 2477,
    "device_type_id": 22,
    "message_type_id": 2236,
    "message_profile_id": 4,
    "message_kind": "MESSAGE_KIND_NORMAL",
    "priority": 4,
    "message_text": {
        "value": "Предоставление доступа на выход"
    },
    "comments": {
        "value": ""
    },
    "utc_offset": null,
    "actual_time": {
        "seconds": "1633035790",
        "nanos": 0
    }
}
}
}

```

- details – дополнительная информация по событию в виде списка пар ключ-значение, где ключ – это код дополнительной информации, привязанной к событию, а значение – это сама дополнительная информация, упакованная в тип *google.protobuf.Any*;
- gid – глобальный идентификатор события;
- id – идентификатор события, в рамках клиентской сессии;
- session_id – идентификатор клиентской сессии;
- server_session_id – идентификатор сессии сервера;
- time – время события;
- source_device_id – идентификатор устройства-источника события;
- device_type_id – идентификатор типа устройства-источника;
- message_type_id – идентификатор типа события;
- message_profile_id – идентификатор профиля события;
- message_kind – вид события. Возможные виды событий:
 - MESSAGE_KIND_UNSPECIFIED – не указан;
 - MESSAGE_KIND_NORMAL – штатные события;
 - MESSAGE_KIND_ALARM – тревожные события;
 - MESSAGE_KIND_FAULT – неисправности;
- priority – приоритет события;
- message_text – текст события;
- comments – комментарий к событию;
- utc_offset – сдвиг часового пояса относительно UTC на момент фиксации события,
- actual_time – время фактического добавления события в базу данных.

Пример запроса получения событий проходов за период с «01.10.2021» по «01.12.2021» сотрудников подразделения с идентификатором 111:

```
{
  "terms": [
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByTimeSearchTerm",
      "value": "ChQKCAjQytiKBhAAEggI0KGajQYQAA=="
    },
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByIsPassageSearchTerm",
      "value": ""
    },
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByPersonParametersSearchTerm",
      "value": "GgMKAW8="
    }
  ],
  "detail_codes": [
    "AttachedPerson"
  ],
  "query_description": {
    "limit": {
      "value": 20
    }
  },
  "include_removed": true
}
```

Где значение фильтра ProtocolMessageByTimeSearchTerm:

```
{
  "time": {
    "from": {
      "seconds": 1633035600,
      "nanos": 0
    },
    "to": {
      "seconds": 1638306000,
      "nanos": 0
    }
  }
}
```

Значение фильтра ProtocolMessageByPersonParametersSearchTerm:

```
{
  "organization_node_ids": {
    "ids": [ 111 ]
  }
}
```

Пример запроса получения всех событий-тревог начиная с последнего идентификатора 34120041:

```
{
  "terms": [
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByLastGlobalIdSearchTerm",
      "value": "C0nCohA="
    },
    {
      "type_url": "type.googleapis.com/
esprom.taurus.grpc.v1.protocol.ProtocolMessageByMessageParametersSearchTerm",
      "value": "GgMKAQA="
    }
  ],
  "include_removed": false
}
```

Где значение фильтра ProtocolMessageByLastGlobalIdSearchTerm:

```
{
  "last_gid": 34120041
}
```

Значение фильтра ProtocolMessageByMessageParametersSearchTerm:

```
{
  "message_kinds": {
    "message_kinds": [ "MessageKind.MESSAGE_KIND_ALARM" ]
  }
}
```

5.24.1.1. Фильтры событий

В фильтрах для записей протокола могут быть использованы стандартные элементы фильтрации, подробное описание которых приведено в [Приложении 3. Основные элементы фильтрации](#).

Файл protocol_info.proto (package esprom.taurus.grpc.v1.protocol)

ProtocolMessageByLastGlobalIdSearchTerm – получение событий из протокола, глобальный идентификатор которых больше, чем указанное значение. Параметры:

- last_gid (int32) – нижняя граница для глобального идентификатора события.

ProtocolMessageByTimeSearchTerm – получение событий, время возникновения которых попадает в указанный диапазон. Параметры:

- time – при получении событий будет выполняться проверка времени события на вхождение в заданный интервал (esprom.taurus.grpc.v1.TimestampSearchTermEntry);

ProtocolMessageByTimeOfDaySearchTerm – получение событий, дневная составляющая времени возникновения которых попадает в указанный диапазон. Параметры:

- time_of_day – при получении событий будет выполняться проверка времени дня события на вхождение в заданный интервал (TimeOfDaySearchTermEntry).

ProtocolMessageByHostInfosSearchTerm – получение событий, источниками возникновения которых являются устройства, принадлежащие заданным серверам оборудования. Параметры:

- host_info_ids – список идентификаторов серверов оборудования (IdsSearchTermEntry).

ProtocolMessageByDriverInstancesSearchTerm – получение событий, источниками возникновения которых являются устройства, относящиеся к указанным экземплярам драйверов. Параметры:

- dins – список идентификаторов экземпляров драйверов (IdsSearchTermEntry).

ProtocolMessageByDeviceTypesSearchTerm – получение событий, источниками возникновения которых являются устройства с заданным типом. Параметры:

- device_types – список кодов типов устройств (IdsSearchTermEntry).

ProtocolMessageBySourceDevicesSearchTerm – получение событий от указанных источников устройств. Параметры:

- sdns – список идентификаторов устройств (IdsSearchTermEntry).

ProtocolMessageByDriverTypesSearchTerm – получение событий, источниками возникновения которых являются устройства, относящиеся к указанным типам драйверов. Параметры:

- driver_ids – список идентификаторов типов драйверов (IdsSearchTermEntry).

ProtocolMessageByMessageParametersSearchTerm – получение событий с фильтрацией по критериям, относящимся к типу события. Параметры:

- message_types – фильтр по типам событий (IdsSearchTermEntry);
- message_profiles – фильтр по профилям событий (IdsSearchTermEntry);
- message_kinds – фильтр по видам событий:
 - message_kinds (repeated esprom.taurus.grpc.v1.messages.MessageKind) – список видов событий;
- priority – фильтр по приоритету событий (Int32SearchTermEntry).

Пример фильтра для получения событий с профилем «СКУД. Отказ в доступе» с приоритетом от 1 до 5:

```
{
  "message_profiles": {
    "ids": [ 45 ]
  },
  "priority": {
    "from": {
      "value": 1
    },
    "to": {
      "value": 5
    }
  }
}
```

- include_access_denied (bool) – включать или нет в ответ на запрос события об отказе в доступе.

ProtocolMessageByActualTimeSearchTerm – получение событий с фильтрацией по времени записи в протокол. Параметры:

- actual_time – описание диапазона времени записи в протокол (TimestampSearchTermEntry).

Пример фильтра для получения событий, записанных в протокол с «01.01.2025» по «01.10.2025»:

```
{
  "actual_time": {
    "from": {
      "seconds": 1735689601,
      "nanos": 0
    },
    "to": {
      "seconds": 1736467201,
      "nanos": 0
    }
  }
}
```

Файл protocol_persons.proto (package esprom.taurus.grpc.v1.protocol)

ProtocolMessageByPersonParametersSearchTerm – получение событий с фильтрацией по параметрам, которые связаны со СКУД:

- person_ids – фильтр по привязанным к событиям персонам (IdsSearchTermEntry);
- organization_node_ids – фильтр по организациям и подразделениям привязанных персон (IdsSearchTermEntry);
- pass_category_ids – фильтр по категории пропусков, привязанных к событиям (IdsSearchTermEntry);
- pass_priority – фильтр по приоритету привязанных пропусков (Int32SearchTermEntry);
- card_code – фильтр по коду идентификации карт, привязанных к событиям (google.protobuf.UInt64Value).

ProtocolMessageByControlAreasSearchTerm – получение событий, источниками возникновения которых являются устройства привязанные к указанным территориям:

- control_area_ids – фильтр по территориям (IdsSearchTermEntry).

5.24.1.2. Дополнительная информация

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
AttachedCameras	Привязанные к событию камеры.	esprom.taurus.grpc.v1.protocol.AttachedCameras	<pre data-bbox="986 562 1460 801"> { "camera_ids": [5351, 5352] }</pre>
AttachedCoordinate	Привязанное к событию географическое положение.	esprom.taurus.grpc.v1.drivers.GeoCoordinate	<pre data-bbox="986 949 1460 1279"> { "altitude": 25, "latitude": 33, "longitude": 53, "horizontal_accuracy": 76, "vertical_accuracy": 98, "speed": 6, "course": 3 }</pre>
AttachedImage	Привязанное к событию изображение.	esprom.taurus.grpc.v1.protocol.AttachedImage	<pre data-bbox="986 1386 1460 1715"> { "image": [67, 58, 47, 85, 115...] }</pre>

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
SourceDevice	Информация об устройстве-источнике события.	esprom.taurus.grpc.v1.drivers.Device	<pre data-bbox="986 488 1463 913"> { "sdn": 6025, "din": 144, "parent_sdn": { "value": 6006 }, "name": "ППЦ - Калитка", "type": 4, "address": 1, "is_active": true, "time_zone_code": 0 } </pre>
MessageType	Информация о типе события.	esprom.taurus.grpc.v1.drivers.MessageType	<pre data-bbox="986 1055 1463 1597"> { "driver_id": 256, "device_type": 4, "code": 1, "text": "Штатный вход %nm %n1", "message_profile_id": 5, "pass_direction": "PASS_DIRECTION_TYPE_IN", "credential_check_result": "CREDENTIAL_CHECK_RESULT_ACCESS _VERIFIED", "credential_check_result_group" : null } </pre>

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
AttachedPerson	Привязанной к событию информация о персоне.	type.googleapis.com/esprom.taurus.grpc.v1.persons.Person	<pre data-bbox="986 488 1465 1211"> { "id": 33696, "name": "Иванов", "first_name": { "value": "Иван" }, "second_name": { "value": "Иванович" }, "table_no": { "value": "000000078298305" }, "comments": null, "organization_node_id": 97, "position_id": { "value": 4616 }, "create_date": { "seconds": 1572620123, "nanos": 0 } } </pre>

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
AttachedPass	Привязанная к событию информация о пропуске.	type.googleapis.com/esprom.taurus.grpc.v1.pers ons.Pass	<pre> { "id": 41852, "person_id": 33696, "pass_category_id": 103, "status": "PASS_STATUS_ACTIVE", "card_id": { "value": 34081 }, "access_level_id": { "value": 14 }, "priority": 1, "create_date": { "seconds": 1347013357, "nanos": 0 }, "activation_time": { "seconds": 1572782465, "nanos": 856021000 }, "issue_date": { "seconds": 1347013468, "nanos": 0 }, "disable_anti_pass_back_check": false } </pre>
AttachedCard	Привязанная к событию информация о карте доступа.	type.googleapis.com/esprom.taurus.grpc.v1.pers ons.Card	<pre> { "id": 34081, "full_card_code": 304958821244, "create_date": { "seconds": 1572620019, "nanos": 0 }, "status": "CARD_STATUS_ISSUED", "identifier_type": 0, "mifare_security_level": 0, "pre_issued": false } </pre>

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
DestinationArea	Информация о территории назначения события прохода (куда был осуществлен вход).	esprom.taurus.grpc.v1.persons.ControlArea	<pre data-bbox="986 488 1465 795"> { "id": 48, "name": "ППЦ", "parent_id": { "value": 2 }, "time_block_id": null } </pre>
SourceArea	Информация об исходной территории события прохода (откуда был осуществлен вход).	esprom.taurus.grpc.v1.persons.ControlArea	<pre data-bbox="986 891 1465 1137"> { "id": 1, "name": "Вне территории", "parent_id": null, "time_block_id": null } </pre>

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
PassCategory	Информация о категории пропуска, привязанного к событию.	esprom.taurus.grpc.v1.persons.PassCategory	<pre data-bbox="986 488 1465 1301"> { "id": 103, "name": "Для служащих", "time_restriction_rule": "TIME_RESTRICTION_RULE_NONE", "time_restriction_unit": "TIME_RESTRICTION_UNIT_DAY", "time_restriction_value": 0, "photo_identification_form_id": { "value": 1 }, "is_change_pass_end_date_allowed": true, "is_pass_prolongation_allowed": true, "numeration_id": null, "inactive_days_count_before_auto_block_pass": null, "activation_time_limit": null } </pre>
AttachedPersonPosition	Информация о должности персоны, привязанной к событию.	esprom.taurus.grpc.v1.persons.DictionaryRecord	<pre data-bbox="986 1402 1465 1680"> { "id": 47814, "header_id": 9, "value": "Машинист бульдозера", "is_system": false } </pre>

Код дополнительной информации, строка	Описание	Тип protobuf-сообщения	Пример
AttachedPersonDepartment	Информация о подразделении персоны, привязанной к событию.	esprom.taurus.grpc.v1.persons.OrganizationNode	<pre>{ "id": 11711, "name": "Московский Офис", "parent_id": { "value": 11710 }, "node_type": "ORGANIZATION_NODE_TYPE_DEPARTMENT" }</pre>
AttachedPassAccessLevel	Информация об уровне доступа пропуска, привязанного к событию.	esprom.taurus.grpc.v1.persons.AccessLevel	<pre>{ "id": 9874, "physical_number": null, "is_weak": false, "name": "Служебный", "main_time_block_id": 0 }</pre>

5.24.2. Метод GetLastProtocolMessageInfo

Запрос на получение информации о последнем записанном в протокол событии. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "gid": {
    "value": 63230991
  }
}
```

- gid – глобальный идентификатор последнего записанного в протокол события.

5.25. AuditService

Файл audit.proto предназначен для получения информации из подсистемы Аудита ПК "Бастион-3".

5.25.1. Метод GetAuditEntries

Запрос на получение данных из журнала аудита системы. Запрос может быть выполнен только с использованием токена доступа.

Параметры запроса :

- `terms` – список запакованных в тип `google.protobuf.Any` фильтров, применяемых к записям журнала аудита. В списке обязательно должен быть указан как минимум один фильтр. Описание поддерживаемых фильтров приведено в Приложении 1 к документу "Бастион-3 – Аудит системы. Руководство оператора".

Результатом будут сообщения с информацией о записях журнала аудита, которые удовлетворяют условиям фильтрации, заданным в параметре `terms`. Пример ответа:

```
{
  "audit_entry": {
    "entry_id": 1,
    "time": {
      "seconds": "1738241060",
      "nanos": 561272000
    },
    "operator_role_id": 7,
    "operator_id": 907,
    "operator_name": {
      "value": "q"
    },
    "client_type": {
      "value": "Admin"
    },
    "client_address": {
      "value": "ESP-SMR-48"
    },
    "operation_type": "AUDIT_OPERATION_TYPE_UPDATE",
    "entity_type": "AuditParams",
    "operation_code": {
      "value": "Updated"
    },
    "description": {
      "value": "Обновление параметров Аудита системы"
    },
    "details": {
      "value": "{\"differences\":{\"IsEnabled\":{\"oldValue\":false,\"newValue\":true},\n\"LogAccess\":{\"oldValue\":false,\"newValue\":true},\n\"LogDepths\":{\"oldValue\":{},\n\"newValue\":\n{\n\"LdapGroup\":1,\n\"OpcGroup\":1,\n\"SnmpGroup\":1,\n\"WebPersonnelGroup\":1,\n\"BiometryGroup\":1,\n\"GraphicsGroup\":1,\n\"SecurityControlGroup\":1,\n\"InspectionGroup\":1,\n\"InformGroup\":1,\n\"MtpMaterialPassesGroup\":1,\n\"MtpTransportPassesGroup\":1,\n\"MessageGroup\":1,\n\"OrgStructureGroup\":1,\n\"ReportGroup\":1,\n\"PersonalInfoGroup\":1,\n\"PersonGroup\":1,\n\"PassGroup\":1,\n\"PcnGroup\":1,\n\"ReplicationChangesGroup\":1,\n\"ReplicationSettingsGroup\":1,\n\"OperatorGroup\":1,\n\"SystemParamsGroup\":1,\n\"DictionaryGroup\":1,\n\"TmkGroup\":1,\n\"ControlAreaGroup\":1,\n\"AttendanceGroup\":1,\n\"AccessLevelGroup\":1,\n\"DeviceGroup\":1,\n\"ReportTemplateGroup\":1}}}"
    },
    "success": null,
    "additional_id1": null,
    "additional_id2": null,
    "additional_id3": null
  }
}
```

```
}

```

- entry_id – уникальный идентификатор записи журнала аудита;
- time – время записи в журнал аудита;
- operator_role_id – идентификатор роли оператора;
- operator_id – идентификатор оператора;
- operator_name – имя оператора;
- client_type – тип клиента (возможные типы приведены в [Таблице 1](#));
- client_address – адрес клиента;
- operation_type – тип операции, соответствующей записи журнала аудита. Возможные типы операций:
 - AUDIT_OPERATION_TYPE_UNSPECIFIED – не указан;
 - AUDIT_OPERATION_TYPE_ADD – добавление;
 - AUDIT_OPERATION_TYPE_UPDATE – изменение;
 - AUDIT_OPERATION_TYPE_DELETE – удаление;
 - AUDIT_OPERATION_TYPE_ACCESS – доступ;
 - AUDIT_OPERATION_TYPE_ACTION – действие;
- entity_type – тип сущности, для которой была сделана запись в журнал аудита;
- operation_code – код операции;
- description – описание записи журнала аудита;
- details – дополнительные детали;
- success – результат действия, если запись журнала аудита соответствует какому-либо действию оператора в системе;
- additional_id1 – дополнительный идентификатор сущности 1;
- additional_id2 – дополнительный идентификатор сущности 2;
- additional_id3 – дополнительный идентификатор сущности 3.

Описание возможных пар типов сущностей и кодов операции приведено в Приложении 1 к документу "Бастион-3 – Аудит системы. Руководство оператора".

Таблица 1. Типы клиентов ПК "Бастион-3" в записях журнала аудита

Строковый код типа клиента	Клиент
Admin	Панель управления
Attendance	Учёт рабочего времени
BAudit	Аудит
Bastion	Пост охраны
BPers	Бюро пропусков

Строковый код типа клиента	Клиент
BRepGen	Отчёт
BWebApi	Доступ к системе через Web API
DriverHost	Сервер оборудования
LocalAgent	Локальный агент
NetCenter	Сервер системы
PCN	ПЦН
Replication	Репликация
StateMonitor	Монитор состояния системы
Undefined	Тип клиента не определён

5.25.1.1. Фильтры записей журнала аудита

При запаковке в тип *google.protobuf.Any* значение поля *type_url* формируется по следующему шаблону: *type.googleapis.com/[имя пакета].[Тип сообщения-фильтра]*, где имя пакета соответствует значению указанному в поле *package* proto-файла, в котором приведено описание фильтра. Например, для фильтра *AuditEntryByOperationTypeSearchTerm*, описанного следующим proto-файлом:

```
syntax = "proto3";

package esprom.taurus.grpc.v1.audit;

message AuditEntryByOperationTypeSearchTerm {
  repeated AuditOperationType operation_types = 1;
}
```

при запаковке в *Any* поле *type_url* получит значение *"type.googleapis.com/еспром.taurus.grpc.v1.audit.AuditEntryByOperationTypeSearchTerm"*.

AuditEntryByOperationTypeSearchTerm – получение записей из журнала аудита, тип операции которых входит в заданное перечисление. Параметры:

- *operation_types* (*repeated esprom.taurus.grpc.v1.audit.AuditOperationType*) – перечисление типов операций.

AuditEntryByLastIdSearchTerm – получение записей из журнала аудита, идентификатор которых больше, чем указанное значение. Параметры:

- *last_id* (*int32*) – нижняя граница для идентификатора записи.

AuditEntryByRoleSearchTerm – получение записей из журнала аудита, связанных с действиями операторов, относящихся к указанным ролям. Параметры:

- `role_ids` (`esprom.taurus.grpc.v1.IdsSearchTermEntry`) – описание фильтра по идентификаторам:
 - `ids` (`repeated int32`) – список идентификаторов ролей операторов.

AuditEntryByOperatorSearchTerm – получение записей из журнала аудита, связанных с заданными операторами. Параметры:

- `operator_ids` (`esprom.taurus.grpc.v1.IdsSearchTermEntry`) – описание фильтра по идентификаторам:
 - `ids` (`repeated int32`) – список идентификаторов операторов.

AuditEntryByChangeDateSearchTerm – получение записей из журнала аудита, которые произошли в заданный интервал времени. Параметры:

- `change_date` (`esprom.taurus.grpc.v1.TimestampSearchTermEntry`) – описание диапазона времени создания записи в журнал (обязательно должна быть указан обе границы),
 - `from` (`google.protobuf.Timestamp`) – нижняя граница времени (включается в результаты запроса);
 - `to` (`google.protobuf.Timestamp`) – верхняя граница времени (не включается в результаты запроса).

AuditEntryByEntityTypeSearchTerm – получение записей из журнала аудита, которые связаны с заданными типами сущностей. Параметры:

- `entity_types` (`esprom.taurus.grpc.v1.StringIdsSearchTermEntry`) – описание фильтра по строковым идентификаторам:
 - `ids` (`repeated string`) – список кодов типов сущностей.

AuditEntryByClientTypeSearchTerm – получение записей из журнала аудита, которые связаны с заданными типами клиентов. Параметры:

- `client_types` (`esprom.taurus.grpc.v1.StringIdsSearchTermEntry`) – описание фильтра по строковым идентификаторам:
 - `ids` (`repeated string`) – список идентификаторов типов клиентов.

5.26. TransportService

Файл `transport.proto` предназначен для получения информации о статусах транспортных средств и транспортных средствах.

5.26.1. Метод GetTransportStatuses

Запрос на получение списка статусов транспорта. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "transport_statuses": [
    {
      "id": 1,
```

```

        "name": "Допуск разрешен"
    },
    {
        "id": 4,
        "name": "Черный список"
    }
]
}

```

Ответное сообщение содержит список всех статусов транспортных средств. О каждом статусе представлена следующая информация:

- id – идентификатор статуса транспорта;
- name – наименование статуса.

5.26.2. Метод GetTransports

Запрос на получение списка транспортных средств. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "search_string": "222"
}

```

- search_string – строка поиска по массиву подстрок, разделенных пробелом, на вхождения которых будут проверяться номера автомобилей, модели и цвета.

Пример ответа на запрос:

```

{
  "transports": [
    {
      "id": 105,
      "registration_number": {
        "value": "2222222222"
      },
      "model_name": null,
      "description": null,
      "manufacture_year": null,
      "color": null,
      "weight": null,
      "volume": null,
      "owner_name": null,
      "transport_type": null,
      "transport_category": "TRANSPORT_CATEGORY_MAINTRANSPORT",
      "transport_status_id": 4
    }
  ]
}

```

Ответное сообщение содержит список всех транспортных средств, удовлетворяющих условию поиска. О каждом транспортном средстве представлена следующая информация:

- id – Идентификатор транспортного средства;
- registration_number – регистрационный номер;
- model_name – модель;
- description – описание;
- manufacture_year – год выпуска;
- color – цвет;
- weight – вес;
- volume – Версия(кузов);
- owner_name – владелец;
- transport_type – тип;
- transport_category – категория транспортного средства:
 - TRANSPORT_CATEGORY_MAINTRANSPORT – основное транспортное средство;
 - TRANSPORT_CATEGORY_SEMITRAILER – полуприцеп;
 - TRANSPORT_CATEGORY_TRAILER – прицеп.
- transport_status_id – ссылка на статус транспортного средства.

Если ни одно транспортное средство не было найдено , в ответ придет пустой список:

```
{
  "transports": []
}
```

5.26.3. Метод GetTransport

Запрос на получение списка транспортного средства по его идентификатору. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "transport_id": 102
}
```

- transport_id – идентификатор транспортного средства.

Пример ответа на запрос:

```
{
  "transport": {
    "id": 102,
    "registration_number": {
      "value": "43355"
    },
    "model_name": null,
  }
}
```

```
    "description": null,  
    "manufacture_year": null,  
    "color": null,  
    "weight": null,  
    "volume": null,  
    "owner_name": null,  
    "transport_type": null,  
    "transport_category": "TRANSPORT_CATEGORY_MAINTRANSPORT",  
    "transport_status_id": 1  
  }  
}
```

Ответное сообщение содержит список всех транспортных средств, удовлетворяющих условию поиска. О каждом транспортном средстве представлена следующая информация:

- id – Идентификатор транспортного средства;
- registration_number – регистрационный номер;
- model_name – модель;
- description – описание;
- manufacture_year – год выпуска;
- color – цвет;
- weight – вес;
- volume – Версия(кузов);
- owner_name – владелец;
- transport_type – тип;
- transport_category – категория транспортного средства:
 - TRANSPORT_CATEGORY_MAINTRANSPORT – основное транспортное средство;
 - TRANSPORT_CATEGORY_SEMITRAILER – полуприцеп;
 - TRANSPORT_CATEGORY_TRAILER – прицеп.
- transport_status_id – ссылка на статус транспортного средства.

5.26.4. Метод GetTransportPhoto

Запрос для получения фото транспортного средства. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{  
  "transport_id": 105  
}
```

- transport_id – идентификатор транспортного средства.

Ответ содержит значение фотографии:

```
{  
  "photo": {
```

```

    "value": "QEBAQEBAQEBAQEBAQE..."
  }
}

```

Если фото у транспортного средства не задано, придет значение null:

```

{
  "photo": null
}

```

5.26.5. Метод GetTransportPhotoHash

Запрос на получение хеша фотографии транспортного средства. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "transport_id": 105
}

```

- transport_id – идентификатор транспортного средства.

Ответ содержит значение хеша фотографии:

```

{
  "hash": {
    "value": "b3ed4a19a4bc556ebffe680e1cdddd61"
  }
}

```

Если фото у транспортного средства не задано, придет значение null:

```

{
  "hash": null
}

```

5.26.6. Метод GetTransportPhotoThumbnail

Запрос на получение превью фотографии транспортного средства. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "transport_id": 105
}

```

```
}

```

- `transport_id` – идентификатор транспортного средства.

Ответ содержит значение хеша фотографии:

```
{
  "thumbnail": {
    "value": "/9j/4AAQSkZJRgABAQAAQABAAD/2wCEAAEBAQE..."
  }
}
```

Если фото у транспортного средства не задано, придет значение `null`:

```
{
  "thumbnail": null
}
```

5.26.7. Метод `TransportChanged`

Запрос на подписку изменения транспортных средств. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "transport_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- `transport_id` – идентификатор транспортного средства, которое изменилось;
- `change_type` – тип изменения транспортного средства:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – транспортное средство создано;
 - `ENTITY_CHANGE_TYPE_UPDATE` – транспортное средство обновлено;
 - `ENTITY_CHANGE_TYPE_DELETE` – транспортное средство удалено.

5.26.8. Метод `TransportPhotoChanged`

Запрос на подписку изменения фотографии транспортного средства. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "transport_id": 101,
```

```
}
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- transport_id – идентификатор транспортного средства, фото которого изменилось;
- change_type – тип изменения фото:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – фото создано;
 - ENTITY_CHANGE_TYPE_UPDATE – фото обновлено;
 - ENTITY_CHANGE_TYPE_DELETE – фото удалено.

5.26.9. Метод TransportStatusChanged

Запрос на подписку изменения статусов транспортных средств. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "transport_status_id": 101,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- transport_status_id – идентификатор статуса транспортного средства, которое изменилось;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – статус транспортного средства создан;
 - ENTITY_CHANGE_TYPE_UPDATE – статус транспортного средства обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – статус транспортного средства удален.

5.27. OperatorsInfoService

Файл operators_info.proto предназначен для получения информации об операторах, их ролях и полномочиях.

5.27.1. Метод GetRoles

Запрос на получение списка ролей операторов. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "include_system_roles": true
}
```

- `include_system_roles` – включать или нет в ответ системные роли.

Пример ответа на запрос:

```
{
  "roles": [
    {
      "id": -66,
      "name": "Система",
      "comments": {
        "value": "Системная роль, используемая для работы служб и сервисов ПК
Бастион-3."
      }
    },
    {
      "id": 7,
      "name": "Администраторы",
      "comments": {
        "value": "Встроенная роль для администраторов системы, обладающая всеми
полномочиями в ПК Бастион-3."
      }
    }
  ]
}
```

Ответное сообщение содержит список всех ролей операторов с учетом фильтра. О каждой роли представлена следующая информация:

- `id` – идентификатор роли оператора;
- `name` – название роли;
- `comments` – комментарий.

5.27.2. Метод `GetOperators`

Запрос на получение списка операторов. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "include_system_operators": false
}
```

- `include_system_operators` – включать или нет в ответ системных операторов.

Пример ответа на запрос:

```
{
  "operators": [
    {
      "id": 1,
      "name": "q",

```

```
        "role_id": 7,  
        "is_active": true,  
        "origin": "OPERATOR_ORIGIN_NATIVE",  
        "login_password": true,  
        "certificate": true,  
        "otp": false  
    }  
]  
}
```

Ответное сообщение содержит список всех операторов с учетом фильтра. О каждом операторе представлена следующая информация:

- id – идентификатор оператора;
- name – имя оператора/логин в системе;
- role_id – идентификатор роли оператора;
- is_active – признак активности учётной записи;
- origin – идентификатор источника данных:
 - OPERATOR_ORIGIN_NATIVE – оператор создан в Бастионе;
 - OPERATOR_ORIGIN_NATIVE_LDAP – оператор пришёл из LDAP;
 - OPERATOR_ORIGIN_NATIVE_OIDC – оператор пришёл из OpenID Connect.
- login_password – признак аутентификации пользователя по паролю;
- certificate – признак аутентификации пользователя по сертификату;
- otp – Признак аутентификации пользователя по секретному слову.

5.27.3. Метод GetOperator

Запрос на получение информации об операторе. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{  
  "operator_id": 1  
}
```

- operator_id – идентификатор оператора.

Пример ответа на запрос:

```
{  
  "operator": {  
    "id": 1,  
    "name": "q",  
    "role_id": 7,  
    "is_active": true,  
    "origin": "OPERATOR_ORIGIN_NATIVE",  
    "login_password": true,  
    "certificate": true,  
    "otp": false  
  }  
}
```

```
}
}
```

Ответное сообщение содержит информацию об операторе. Представлена следующая информация:

- id – идентификатор оператора;
- name – имя оператора/логин в системе;
- role_id – идентификатор роли оператора;
- is_active – признак активности учётной записи;
- origin – идентификатор источника данных:
 - OPERATOR_ORIGIN_NATIVE – оператор создан в Бастионе;
 - OPERATOR_ORIGIN_NATIVE_LDAP – оператор пришёл из LDAP;
 - OPERATOR_ORIGIN_NATIVE_OIDC – оператор пришёл из OpenID Connect.
- login_password – признак аутентификации пользователя по паролю;
- certificate – признак аутентификации пользователя по сертификату;
- otp – Признак аутентификации пользователя по секретному слову.

5.27.4. Метод GetRole

Запрос на получение роли по идентификатору. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "role_id": 1
}
```

- role_id – идентификатор роли оператора.

Пример ответа на запрос:

```
{
  "role": {
    "id": 7,
    "name": "Администраторы",
    "comments": {
      "value": "Встроенная роль для администраторов системы, обладающая всеми полномочиями в ПК Бастион-3."
    }
  }
}
```

Ответное сообщение содержит список роль оператора. Представлена следующая информация:

- id – идентификатор роли оператора;
- name – название роли;
- comments – комментарий.

5.27.5. Метод GetOperatorInfos

Запрос на получение списка персональных атрибутов операторов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "operatorInfos": [
    {
      "operator_id": 0,
      "full_name": {
        "value": "Система"
      },
      "position": null,
      "table_no": null,
      "email": null,
      "phone": null,
      "telegram": null,
      "skype": null,
      "comments": {
        "value": "Системный оператор, от имени которого выполняются автоматические
        действия."
      }
    },
    ...
  ]
}
```

Ответное сообщение содержит список персональных атрибутов операторов. О каждом операторе представлена следующая информация:

- operator_id – идентификатор оператора;
- full_name – ФИО;
- position – должность;
- table_no – табельный номер;
- email – адрес электронной почты;
- phone – телефон оператора;
- telegram – адрес в Telegram;
- skype – адрес в Skype;
- comments – комментарии.

5.27.6. Метод GetOperatorInfo

Запрос на получение информации о персональных атрибутах оператора. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "operator_id": 1
}
```

```
}

```

- operator_id – идентификатор оператора.

Пример ответа на запрос:

```
{
  "operator_info": {
    "operator_id": 1,
    "full_name": {
      "value": "Иванов И.И."
    },
    "position": null,
    "table_no": null,
    "email": null,
    "phone": null,
    "telegram": null,
    "skype": null,
    "comments": null
  }
}
```

Ответное сообщение содержит информацию о персональных атрибутах оператора. Представлена следующая информация:

- operator_id – идентификатор оператора;
- full_name – ФИО;
- position – должность;
- table_no – табельный номер;
- email – адрес электронной почты;
- phone – телефон оператора;
- telegram – адрес в Telegram;
- skype – адрес в Skype;
- comments – комментарии.

5.27.7. Метод ChangePassword

Запрос на смену пароля текущего авторизованного оператора. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "old_password": "q",
  "new_password": "q1"
}
```

После выполнения запроса происходит потеря подключения к серверу системы, для дальнейшей работы требуется повторное подключение под новым паролем.

5.27.8. Метод RoleChanged

Запрос на подписку изменения ролей операторов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "role_id": 100,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- role_id – идентификатор роли оператора, которая изменилась;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – роль оператора создана;
 - ENTITY_CHANGE_TYPE_UPDATE – роль оператора обновлена;
 - ENTITY_CHANGE_TYPE_DELETE – роль оператора удалена.

5.27.9. Метод OperatorChanged

Запрос на подписку изменения операторов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "operator_id": 100,
  "change_type": "ENTITY_CHANGE_TYPE_ADD"
}
```

- operator_id – идентификатор оператора, который изменился;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – оператор создан;
 - ENTITY_CHANGE_TYPE_UPDATE – оператор обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – оператор удален.

5.27.10. Метод GetGeneralDevicePermissions

Запрос на получение разрешений ролей операторов для устройств. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "role_id": {
```

```

    "value": 103
  },
  "devices_filter": {
    "ids": [200, 204]
  }
}

```

- role_id – идентификатор роли оператора, для которой запрашиваются разрешения;
- devices_filter – перечисление идентификаторов устройств, для которых запрашиваются разрешения.

В ответ будут получено сообщение с информацией об общесистемных правах доступа к указанным устройствам:

```

{
  "general_device_permissions": [
    {
      "role_id": 103,
      "sdn": 204,
      "can_receive_message": true,
      "can_confirm_message": true,
      "can_execute_command": true
    }
  ]
}

```

- role_id – идентификатор роли оператора;
- sdn – идентификатор устройства;
- can_receive_message – разрешение на получение/просмотр сообщений;
- can_confirm_message – разрешение на подтверждение тревожных сообщений;
- can_execute_command – разрешение на выполнение команд, действий.

5.27.11. Метод GetRestrictedDeviceActions

Запрос на получение запрещённых для устройств действий/команд. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "role_id": {
    "value": 110
  },
  "devices_filter": {
    "ids": [1991, 1992, 2010]
  }
}

```

- role_id – идентификатор роли оператора, для которой запрашиваются запрещенные действия;

- `devices_filter` – перечисление идентификаторов устройств, для которых запрашиваются запрещенные действия.

В ответ будут получены сообщения с информацией о дополнительно запрещённых для указанной роли действиях для заданных устройств:

```
{
  "restricted_device_actions": {
    "restricted_action_codes": [
      45
    ],
    "role_id": 110,
    "sdn": 2010
  }
}
```

- `restricted_action_codes` – перечисление специфичных кодов действий (команд), запрещенных для выполнения;
- `role_id` – идентификатор роли оператора;
- `sdn` – идентификатор устройства.

5.27.12. Метод `GeneralDevicePermissionsChanged`

Запрос на отслеживание изменений общесистемных прав доступа к устройствам для роли оператора. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример сообщения:

```
{
  "role_id": 117,
  "sdns": [
    1778,
    1780,
    1794
  ]
}
```

- `role_id` – уникальный идентификатор роли оператора, для которой были изменены общесистемные права доступа к устройствам;
- `sdns` – перечисление идентификаторов устройств, к которым были изменены общесистемные права доступа.

5.27.13. Метод `RestrictedDeviceActionsChanged`

Запрос на отслеживание изменений списков запрещённых для устройств команд/действий для роли оператора. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример сообщения:

```
{
  "role_id": 120,
  "sdns": [
    1808,
    1821,
    1828
  ]
}
```

- role_id – уникальный идентификатор роли оператора, для которой были изменены списки запрещённых для устройств команд/действий;
- sdns – перечисление идентификаторов устройств, для которых были изменены списки запрещённых команд/действий.

5.27.14. Метод GetAllBastionSettings

Запрос на получение настроек Поста охраны для всех ролей операторов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "role_bastion_settings": [
    {
      "role_id": 7,
      "map_set_id": {
        "value": 1
      },
      "ask_alarm_reason_on_confirm": false,
      "show_normal_message": true,
      "show_fault_message": true,
      "show_alarm_message": true,
      "check_message_priority_on_show": true,
      "min_message_priority_to_show": 2,
      "map_auto_switch_enabled": false,
      "check_message_priority_on_switch_map": false,
      "min_message_priority_to_switch_map": 0
    }
  ]
}
```

Ответное сообщение содержит информацию о персональных атрибутах оператора. Представлена следующая информация:

- ask_alarm_reason_on_confirm – признак подтверждения тревожных сообщений;
- show_normal_message – признак отображения нормальных сообщений;
- show_fault_message – признак отображения нештатных сообщений;
- show_alarm_message – признак отображения тревожных сообщений;
- check_message_priority_on_show – признак проверки приоритета сообщения;
- min_message_priority_to_show – минимальный приоритет для отображения сообщения;
- map_auto_switch_enabled – признак автоматического переключения между планами;

- `check_message_priority_on_switch_map` – признак проверки приоритета при переключении планов;
- `min_message_priority_to_switch_map` – минимальный приоритет для переключения между планами;

5.27.15. Метод `GetRoleBastionSettings`

Запрос на получение настроек роли для Поста Охраны. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "role_id": 100
}
```

- `role_id` – идентификатор роли оператора, для которой запрашиваются настройки.

Пример ответа:

```
{
  "role_bastion_settings": {
    "role_id": 100,
    "map_set_id": {
      "value": 1
    },
    "ask_alarm_reason_on_confirm": false,
    "show_normal_message": true,
    "show_fault_message": true,
    "show_alarm_message": true,
    "check_message_priority_on_show": true,
    "min_message_priority_to_show": 2,
    "map_auto_switch_enabled": false,
    "check_message_priority_on_switch_map": false,
    "min_message_priority_to_switch_map": 0
  }
}
```

- `role_id` – идентификатор роли;
- `map_set_id` – идентификатор набора планов;
- `ask_alarm_reason_on_confirm` – признак подтверждения тревожных сообщений;
- `show_normal_message` – признак отображения нормальных сообщений;
- `show_fault_message` – признак отображения нештатных сообщений;
- `show_alarm_message` – признак отображения тревожных сообщений;
- `check_message_priority_on_show` – признак проверки приоритета сообщения;
- `min_message_priority_to_show` – минимальный приоритет для отображения сообщения;
- `map_auto_switch_enabled` – признак автоматического переключения между планами;

- `check_message_priority_on_switch_map` – признак проверки приоритета при переключении планов;
- `min_message_priority_to_switch_map` – минимальный приоритет для переключения между планами;

5.27.16. Метод `GetApplicationPermissions`

Запрос на получение прав доступа к функциям приложений для ролей операторов. В качестве фильтра может быть задан идентификатор роли оператора, тогда будут возвращены данные только для заданной роли. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "role_id": {
    "value": 7
  }
}
```

`role_id` – идентификатор роли оператора, для которой запрашиваются запрещенные действия;

Пример ответа на запрос:

```
{
  "application_permissions": [
    {
      "role_id": 7,
      "permission_code": "Persons.ChangePassEndDate"
    },
    {
      "role_id": 7,
      "permission_code": "Persons.EditCatalog.BlockReason"
    }
  ]
}
```

Ответное сообщение содержит информацию о функциональных полномочиях для ролей операторов. Представлена следующая информация:

- `role_id` – идентификатор роли;
- `permission_code` – строковый код полномочия;

5.27.17. Метод `CheckDevicePermission`

Запрос на выполнение проверки на наличие у роли права выполнить действие для устройства. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "action_code": 3,
```

```

    "role_id": 7,
    "sdn": 5
  }

```

- action_code – код действия;
- role_id – идентификатор роли;
- sdn – идентификатор устройства.

Пример ответа на запрос:

```

{
  "has_permission": true
}

```

Ответное сообщение содержит информацию о наличии или отсутствии разрешения.

5.27.18. Метод CheckApplicationPermission

Запрос на выполнение проверки на наличие у роли оператора доступа к функции приложения. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "permission_code": "Main.AtdLaunch",
  "role_id": 101
}

```

- permission_code – строковый код функции приложения;
- role_id – идентификатор роли.

Пример ответа на запрос:

```

{
  "has_permission": false
}

```

Ответное сообщение содержит информацию о наличии или отсутствии разрешения.

5.27.19. Метод ApplicationPermissionsChanged

Запрос на подписку об изменении прав доступа к функциям приложений для роли оператора. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "role_id": 100
}
```

- role_id – идентификатор роли, которая изменилась.

5.27.20. Метод RoleBastionSettingsChanged

Запрос на подписку об изменении параметров Поста Охраны для роли оператора. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "role_id": 100
}
```

- role_id – идентификатор роли, которая изменилась.

5.28. RegimeService

Файл regime.proto предназначен для получения информации о типах рабочих дней и рабочим дням.

5.28.1. Метод GetWorkDayTypes

Запрос на получение списка типов рабочих дней. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "work_day_types": [
    {
      "wdt_code": "Д0",
      "name": "Отпуск без сохранения заработной платы, предоставляемый работнику по разрешению работодателя",
      "color": 14599344,
      "use_in_regimes": false,
      "use_in_special_days": false,
      "numeric_code": "16",
      "is_not_absences": false
    },
    ...
  ]
}
```

Ответное сообщение содержит список всех типов рабочих дней. О каждом типе представлена следующая информация:

- wdt_code – идентификатор типа рабочего дня;
- name – название;
- color – цвет;
- use_in_regimes – признак использования в рабочих днях/сменах;
- use_in_special_days – признак использования в специальных днях;
- numeric_code – числовой код;
- is_not_absences – признак того, что тип рабочего дня не является неявкой (неявки отображаются в Т-13 для расчетного отдела).

5.28.2. Метод GetRegimes

Запрос на получение списка рабочих дней. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "regimes": [
    {
      "id": 621,
      "name": "Нормальный 1 смена",
      "short_name": {
        "value": "Н1"
      },
      "description": null,
      "color": 12180223,
      "delay_start": {
        "seconds": "-2209003200",
        "nanos": 0
      },
      "delay_end": {
        "seconds": "-2209003200",
        "nanos": 0
      },
      "wdt_code": "Я",
      "regime_norm": {
        "seconds": "-2208974400",
        "nanos": 0
      },
      "regime_start": {
        "seconds": "-2208974400",
        "nanos": 0
      },
      "regime_end": {
        "seconds": "-2208942000",
        "nanos": 0
      },
      "is_simple_mode": false,
      "break_start": {
        "seconds": "-2208960000",
        "nanos": 0
      },
      "break_end": {
        "seconds": "-2208956400",
        "nanos": 0
      }
    }
  ]
}
```

```
    },
    "calc_night_time": false,
    "calc_over_time": false,
    "early_in_time": null,
    "late_out_time": null,
    "road_time": null,
    "break_start2": null,
    "break_end2": null,
    "regime_start_in_prior_day": false
  },
  ...
]
```

Ответное сообщение содержит список всех рабочих дней. О каждом рабочем дне представлена следующая информация:

- id – идентификатор рабочего дня;
- name – название;
- short_name – сокращенное название;
- description – описание;
- color – цвет;
- delay_start – погрешность прихода (время, не считающееся опозданием);
- delay_end – погрешность ухода (время, не считающееся ранним уходом);
- wdt_code – идентификатор типа рабочего дня;
- regime_norm – норма отработанного времени;
- regime_start – время начала рабочего дня;
- regime_end – время окончания рабочего дня;
- is_simple_mode – признак принадлежности рабочего дня к упрощенному режиму расчета рабочего времени;
- break_start – время начала перерыва;
- break_end – время окончания перерыва;
- calc_night_time – признак необходимости считать ночное время;
- calc_over_time – признак необходимости считать переработки;
- early_in_time – допустимый ранний приход (входит в РВ);
- late_out_time – допустимый поздний уход (входит в РВ);
- road_time – ходовое время (вычитается из РВ при входе и выходе);
- break_start2 – время начала дополнительного перерыва для отдыха и питания;
- break_end2 – время окончания дополнительного перерыва для отдыха и питания;
- regime_start_in_prior_day – признак начала рабочего дня в предыдущих сутках.

5.28.3. Метод RegimeChanged

Запрос на подписку изменения рабочих дней. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "regime_id": 10004,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- regime_id – идентификатор рабочего дня, который изменился;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – рабочий день создан;
 - ENTITY_CHANGE_TYPE_UPDATE – рабочий день обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – рабочий день удален.

5.29. ShiftScheduleService

Файл shift_schedule.proto предназначен для получения информации о графиках сменности и позициях графика сменности.

5.29.1. Метод GetShiftSchedules

Запрос на получение списка всех графиков сменности. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "shift_schedules": [
    {
      "id": 1,
      "name": "Пятидневная неделя 8:00 - 17:00",
      "description": {
        "value": "Пятидневка"
      },
      "is_default": true,
      "use_holidays": false
    },
    ...
  ]
}
```

Ответное сообщение содержит список всех графиков сменности. О каждом графике представлена следующая информация:

- id – идентификатор графика сменности;
- name – название;
- description – описание;
- is_default – признак значения по умолчанию;
- use_holidays – признак необходимости учитывать праздники.

5.29.2. Метод GetShiftScheduleItems

Запрос на получение списка всех позиций графиков сменности. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "shift_schedule_items": [
    {
      "id": 1,
      "shift_schedule_id": 1,
      "regime_id": 621,
      "day_number": 1
    },
    ...
  ]
}
```

Ответное сообщение содержит список всех позиций графиков сменности. О каждой позиции графика сменности представлена следующая информация:

- id – идентификатор позиции графика сменности;
- shift_schedule_id – идентификатор графика сменности;
- regime_id – идентификатор рабочего дня;
- day_number – номер дня в графике сменности.

5.29.3. Метод GetShiftScheduleItemsForShiftSchedule

Запрос на получение списка всех позиций графиков сменности для конкретного графика. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "shift_schedule_id": 1
}
```

- shift_schedule_id – идентификатор графика сменности.

Пример ответа на запрос:

```
{
  "shift_schedule_items": [
    {
      "id": 1,
      "shift_schedule_id": 1,
      "regime_id": 621,
      "day_number": 1
    },
    ...
  ]
}
```

```
}

```

Ответное сообщение содержит список всех позиций графиков сменности для конкретного графика. О каждой позиции графика сменности представлена следующая информация:

- id – идентификатор позиции графика сменности;
- shift_schedule_id – идентификатор графика сменности;
- regime_id – идентификатор рабочего дня;
- day_number – номер дня в графике сменности.

5.29.4. Метод ShiftScheduleChanged

Запрос на подписку изменения графиков сменности. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "shift_schedule_id": 103,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- shift_schedule_id – идентификатор графика сменности, который изменился;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – график сменности создан;
 - ENTITY_CHANGE_TYPE_UPDATE – график сменности обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – график сменности удален.

5.29.5. Метод ShiftScheduleItemChanged

Запрос на подписку изменения позиций графиков сменности. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "shift_schedule_item_id": 116,
  "new_regime_id": {
    "value": 2763
  },
  "old_regime_id": {
    "value": 621
  },
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- shift_schedule_item_id – идентификатор позиции графика сменности, которая изменилась;

- `old_regime_id` – идентификатор рабочего дня до редактирования;
- `new_regime_id` – идентификатор рабочего дня после редактирования;
- `change_type` – тип изменения:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – позиция графика сменности создана;
 - `ENTITY_CHANGE_TYPE_UPDATE` – позиция графика сменности обновлена;
 - `ENTITY_CHANGE_TYPE_DELETE` – позиция графика сменности удалена.

5.29.6. Метод `GetShiftScheduleOrganizations`

Запрос на получение списка всех настроек признаков значений по умолчанию графиков сменности для организаций/подразделений. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "shift_schedule_organizations": [
    {
      "shift_schedule_id": 121,
      "organization_structure_id": 2
    },
    ...
  ]
}
```

Ответное сообщение содержит список всех настроек признаков значений по умолчанию графиков сменности для организаций/подразделений. В нем представлена следующая информация:

- `shift_schedule_id` – идентификатор графика сменности;
- `organization_structure_id` – идентификатор организации/подразделения.

5.30. `WorkContractService`

Файл `work_contract.proto` предназначен для получения информации о трудовых договорах.

5.30.1. Метод `GetWorkContract`

Запрос на получение трудового договора идентификатору. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "work_contract_id": 751
}
```

- `work_contract_id` – идентификатор трудового договора.

Пример ответа на запрос:

```
{
  "work_contract": {
    "id": 751,
    "start_date": {
      "seconds": "1743278400",
      "nanos": 0
    },
    "end_date": {
      "seconds": "1742760000",
      "nanos": 0
    },
    "shift_schedule_id": 1,
    "shift_schedule_item_day_number": 4,
    "post_name": {
      "value": "Уборщик производственных помещений"
    },
    "organization_structure_id": {
      "value": 142
    },
    "person_id": 104,
    "calc_method_id": 4
  }
}
```

Ответное сообщение содержит данные по трудовому договору. В них содержится следующая информация:

- id – идентификатор трудового договора;
- start_date – дата начала действия;
- end_date – дата окончания действия;
- shift_schedule_id – идентификатор графика сменности;
- shift_schedule_item_day_number – номер первого рабочего дня в графике сменности;
- post_name – должность;
- organization_structure_id – идентификатор организации/подразделения;
- person_id – идентификатор работника;
- calc_method_id – идентификатор метода расчета рабочего времени.

5.30.2. Метод GetWorkContractsForPerson

Запрос на получение трудовых договоров для работника. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_id": 737
}
```

- `person_id` – идентификатор работника.

Пример ответа на запрос:

```
{
  "work_contracts": [
    {
      "id": 1335,
      "start_date": {
        "seconds": "1320955200",
        "nanos": 0
      },
      "end_date": null,
      "shift_schedule_id": 1,
      "shift_schedule_item_day_number": 1,
      "post_name": null,
      "organization_structure_id": null,
      "person_id": 737,
      "calc_method_id": 2
    }
  ]
}
```

Ответное сообщение содержит список всех трудовых договоров для работника. О каждой позиции трудового договора представлена следующая информация:

- `id` – идентификатор трудового договора;
- `start_date` – дата начала действия;
- `end_date` – дата окончания действия;
- `shift_schedule_id` – идентификатор графика сменности;
- `shift_schedule_item_day_number` – номер первого рабочего дня в графике сменности;
- `post_name` – должность;
- `organization_structure_id` – идентификатор организации/подразделения;
- `person_id` – идентификатор работника;
- `calc_method_id` – идентификатор метода расчета рабочего времени.

5.30.3. Метод `GetPersonCountWithoutWorkContract`

Запрос на подсчет количества работников без трудовых договоров. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "person_count": 3
}
```

- `person_count` – количество работников без трудовых договоров.

5.30.4. Метод WorkContractsCreate

Запрос на запуск процедуры создания трудовых договоров для работников, у которых их нет. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "work_contract_count": 3
}
```

- work_contract_count – количество созданных трудовых договоров.

5.30.5. Метод WorkContractChanged

Запрос на подписку изменения трудовых договоров. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
  "work_contract_id": 1322,
  "new_shift_schedule_id": {
    "value": 103
  },
  "old_shift_schedule_id": {
    "value": 1
  },
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- work_contract_id – идентификатор трудового договора, который изменился;
- old_shift_schedule_id – идентификатор графика сменности до изменения;
- new_shift_schedule_id – идентификатор графика сменности после изменения;
- change_type – тип изменения:
 - ENTITY_CHANGE_TYPE_UNSPECIFIED – тип изменения не указан;
 - ENTITY_CHANGE_TYPE_ADD – трудовой договор создан;
 - ENTITY_CHANGE_TYPE_UPDATE – трудовой договор обновлен;
 - ENTITY_CHANGE_TYPE_DELETE – трудовой договор удален.

5.30.6. Метод GetCalcMethods

Запрос на получение списка всех доступных методов расчета рабочего времени. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "calc_methods": [
    {
      "id": 1,
      "name": "1. Последний выход минус первый вход"
    },
    ...
  ]
}
```

Ответное сообщение содержит список всех доступных методов расчета рабочего времени. О каждом методе расчета представлена следующая информация:

- `id` – идентификатор метода расчета;
- `name` – название метода расчета.

5.30.7. Метод `GetShiftScheduleId`

Запрос на получение графика сменности по умолчанию для работника. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_id": 737
}
```

- `person_id` – идентификатор работника.

Пример ответа на запрос:

```
{
  "shift_schedule_id": 1
}
```

- `shift_schedule_id` – идентификатор графика сменности.

5.30.8. Метод `GetAllPassCategoryAtdSettings`

Запрос на получение списка настроек параметров учета рабочего времени для всех категорий пропусков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "pass_category_atd_settings": [
    {
      "pass_category_id": 3,
      "use_in_wtp": true
    }
  ]
}
```

```

        },
        ...
    ]
}

```

Ответное сообщение содержит список настроек параметров учета рабочего времени для всех категорий пропусков. В нем представлена следующая информация:

- `pass_category_id` – идентификатор категории пропуска;
- `use_in_wtp` – признак необходимости вести учёт рабочего времени.

5.31. PassAdditionalFieldService

Файл `pass_additional_field.proto` предназначен для получения информации о дескрипторах дополнительных полей пропуска и значениях дополнительных полей пропусков.

5.31.1. Метод GetPassAdditionalFieldDescriptor

Запрос на получение информации о дескрипторе. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "descriptor_id": 1
}

```

- `descriptor_id` – идентификатор дескриптора;

Ответ:

```

{
  "field_descriptor": {
    "id": 1,
    "name": "стаж",
    "description": null,
    "is_allowed": true
  }
}

```

- `id` – идентификатор дескриптора;
- `name` – наименование дескриптора;
- `description` – назначение дескриптора.

5.31.2. Метод GetPassAdditionalFieldDescriptors

Запрос на получение полного списка дескрипторов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ содержит список дескрипторов в системе:

```
{
  "descriptors": [
    {
      "id": 1,
      "name": "стаж",
      "description": null,
      "is_allowed": true
    },
    {
      "id": 2,
      "name": "резервный адрес",
      "description": null,
      "is_allowed": true
    },
    ...
  ]
}
```

5.31.3. Метод GetPassAdditionalFieldValues

Запрос для получения значений дополнительных полей. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса, если нужно получить значения для одного пропуска:

```
{
  "by_pass": {
    "pass_id": 101
  }
}
```

- pass_id – идентификатор пропуска;

Если необходимо получить значения для нескольких пропусков, используется следующий запрос, где необходимо перечислить идентификаторы пропусков:

```
{
  "by_passes": {
    "pass_ids": [
      101,
      365
    ]
  }
}
```

Результатом запроса будет список значений дополнительных полей:

```
{
  "values": [
```

```

    {
      "pass_id": 101,
      "descriptor_id": 3,
      "boolean": true
    },
    {
      "pass_id": 101,
      "descriptor_id": 4,
      "integer": 12345
    }
  ]
}

```

- pass_id – идентификатор пропуска;
- descriptor_id – идентификатор дескриптора;
- Следующее поле может отличаться в зависимости от того, какой тип значений указан в дескрипторе:
 - boolean – булево значение;
 - integer – целое числовое;
 - float – дробное число;
 - string – текст;
 - date_time – дата и время;
 - date – дата;
 - time_span – время.

5.31.4. Метод GetPassCategoryTypedAdditionalFields

Запрос для получения дескрипторов, доступных для задания у категории пропуска. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "category_id": 1
}

```

- category_id – идентификатор категории пропуска;

Результатом запроса будет список дескрипторов, доступных для задания для категории пропуска, вместе со значениями по умолчанию:

```

{
  "values": [
    "descriptor_id": 119,
    "pass_category_id": 1,
    "is_required": true,
    "integer": {
      "value": "999999"
    }
  ]
}

```

```
}

```

- `descriptor_id` – идентификатор дескриптора;
- `pass_category_id` – идентификатор категории пропуска;
- `is_required` – признак, указывающий на необходимость обязательного заполнения поля значением;
- Одно из следующих полей содержит значение по умолчанию:
 - `boolean` – булево значение;
 - `integer` – целое числовое;
 - `float` – дробное число;
 - `string` – текст;
 - `date_time` – дата и время;
 - `date` – дата;
 - `time_span` – время.

5.31.5. Метод `PassAdditionalFieldDescriptorChanged`

Запрос на подписку об изменениях дескрипторов дополнительных полей пропусков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "descriptor_id": 131,
  "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
}
```

- `descriptor_id` – идентификатор дескриптора, который изменился;
- `change_type` – тип изменения:
 - `ENTITY_CHANGE_TYPE_UNSPECIFIED` – тип изменения не указан;
 - `ENTITY_CHANGE_TYPE_ADD` – дескриптор создан;
 - `ENTITY_CHANGE_TYPE_UPDATE` – дескриптор обновлен;
 - `ENTITY_CHANGE_TYPE_DELETE` – дескриптор удален.

5.31.6. Метод `PassAdditionalFieldValueChanged`

Запрос на подписку об изменениях значений дополнительных полей пропусков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "value": {
    "pass_id": 103,
    "descriptor_id": 119,
    "integer": "123123"
  }
}
```

```

    },
    "change_type": "ENTITY_CHANGE_TYPE_UPDATE"
  }

```

- pass_id – идентификатор пропуска;
- descriptor_id – идентификатор дескриптора;
- Одно из следующих полей содержит новое значение дескриптора:
 - boolean – булево значение;
 - integer – целое числовое;
 - float – дробное число;
 - string – текст;
 - date_time – дата и время;
 - date – дата;
 - time_span – время;
- change_type – тип изменения.

5.32. DeviceControlAreaLinkService

Файл device_control_area_link.proto предназначен для получения информации о привязке устройств к территориям.

5.32.1. Метод GetDeviceControlAreaLinks

Запрос на получение списка явно заданных привязок устройств к территориям. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ содержит список явно заданных привязок устройств к территориям:

```

{
  "device_control_area_links": [
    {
      "sdn": 202,
      "control_area_id": 0
    },
    {
      "sdn": 209,
      "control_area_id": 2
    }
  ]
}

```

- sdn – идентификатор устройства;
- control_area_id – идентификатор территории, к которой устройство привязано.

Все дочерние устройства от полученных в ответе устройств наследуют значение привязки к территории от своего родителя.

5.32.2. Метод GetLinkedDevicesByControlAreald

Запрос на получение списка устройств, привязанных к территории. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "control_area_id": 2,
  "include_inherited": true
}
```

- control_area_id – идентификатор территории;
- include_inherited – флаг, отвечающий за включение в результат устройств, которые наследуют значение привязки к территории от родительского устройства.

Ответ содержит список устройств, привязанных к указанной территории:

```
{
  "devices": [
    {
      "sdn": 234,
      "din": 100,
      "parent_sdn": {
        "value": 208
      },
      "name": "Дверь 1.1.2",
      "type": 3,
      "address": 2,
      "is_active": true,
      "time_zone_code": 0
    },
    {
      "sdn": 235,
      "din": 100,
      "parent_sdn": {
        "value": 234
      },
      "name": "Дверь 1.1.2 R3",
      "type": 19,
      "address": 3,
      "is_active": true,
      "time_zone_code": 0
    },
    {
      "sdn": 236,
      "din": 100,
      "parent_sdn": {
        "value": 234
      },
      "name": "Дверь 1.1.2 R4",
      "type": 19,
      "address": 4,
      "is_active": true,
      "time_zone_code": 0
    }
  ]
}
```

```
]
}
```

5.32.3. Метод GetLinkedControlAreaTo

Запрос для получения территории, к которой привязано устройство. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "sdn": 234
}
```

- sdn – идентификатор устройства.

Результатом запроса будет территория, к которой привязано устройство:

```
{
  "control_area": {
    "id": 2,
    "name": "На территории",
    "parent_id": null,
    "time_block_id": null
  }
}
```

5.33. BiometryService

Файл biometry.proto предназначен для получения информации о биометрических шаблонах для персон.

5.33.1. Метод GetBioTemplateTypes

Запрос на получение списка типов биометрических шаблонов. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "bio_template_types": [
    {
      "code": 201,
      "name": {
        "value": "Указательный палец правой руки"
      },
      "biometry_type": "BIOMETRY_TYPE_FINGERPRINT",
      "driver_type_id": 274,
      "bio_reader_type_id": 5
    },
  ],
}
```

```
{
  "code": 202,
  "name": {
    "value": "Средний палец правой руки"
  },
  "biometry_type": "BIOMETRY_TYPE_FINGERPRINT",
  "driver_type_id": 274,
  "bio_reader_type_id": 5
},
{
  "code": 203,
  "name": {
    "value": "Безымянный палец правой руки"
  },
  "biometry_type": "BIOMETRY_TYPE_FINGERPRINT",
  "driver_type_id": 274,
  "bio_reader_type_id": 5
}
]
```

- code – код биометрического шаблона;
- name – имя биометрического шаблона;
- biometry_type – тип биометрии:
 - BIOMETRY_TYPE_FINGERPRINT – отпечаток пальца;
 - BIOMETRY_TYPE_FACIAL_IMAGE – геометрия/изображение лица;
 - BIOMETRY_TYPE_PALM_VEINS – рисунок вен ладони.
- driver_type_id – идентификатор типа драйвера;
- bio_reader_type_id – идентификатор типа биометрического считывателя.

5.33.2. Метод GetPersonBioTemplateInfos

Запрос на получение списка информации о биометрических шаблонах для персоны. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_id": 149
}
```

- person_id – идентификатор персоны, для которой запрашивается информация о биометрических шаблонах.

В качестве результата получаем список информации о биометрических шаблонах:

```
{
  "bio_template_infos": [
    {
      "id": 13,
      "bio_template_code": 1
    }
  ]
}
```

```

    },
    {
      "id": 12,
      "bio_template_code": 10
    },
    {
      "id": 11,
      "bio_template_code": 5
    }
  ]
}

```

- id – уникальный идентификатор биометрического шаблона;
- bio_template_code – код типа биометрического шаблона.

5.33.3. Метод GetPersonBioTemplates

Запрос на получение списка биометрических шаблонов для персоны. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "person_id": 149
}

```

- person_id – идентификатор персоны, для которой запрашиваются биометрические шаблоны.

В качестве результата получаем список биометрических шаблонов:

```

{
  "bio_templates": [
    {
      "id": 13,
      "person_id": 149,
      "bio_template_code": 1,
      "data": "QEBAQEBAQEBAQEBAQE..."
    },
    {
      "id": 12,
      "person_id": 149,
      "bio_template_code": 10,
      "data": "QEBAQEBAQEBAQEBAQE..."
    },
    {
      "id": 11,
      "person_id": 149,
      "bio_template_code": 5,
      "data": "QEBAQEBAQEBAQEBAQE..."
    }
  ]
}

```

- id – идентификатор биометрического шаблона;
- person_id – идентификатор персоны;
- bio_template_code – код типа биометрического шаблона;
- data – цифровое представление биометрического шаблона.

5.33.4. Метод `GetPersonsBioTemplates`

Запрос на получение списка биометрических шаблонов для списка персон. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "person_ids": [149, 150],
  "bio_template_codes_filter": {
    "codes": [5]
  }
}
```

- person_ids – список идентификаторов персон, для которых запрашиваются биометрические шаблоны;
- bio_template_codes_filter – список кодов биометрических шаблонов.

Ответное сообщение содержит информацию о биометрических шаблонах, которые подошли под указанные фильтры:

```
{
  "bio_template": {
    "id": 149,
    "person_id": 54992,
    "bio_template_code": 5,
    "data": "QEBAQEBAQEBAQEBAQE..."
  }
}
```

- id – идентификатор биометрического шаблона;
- person_id – идентификатор персоны;
- bio_template_code – код типа биометрического шаблона;
- data – цифровое представление биометрического шаблона.

5.33.5. Метод `PersonBioTemplatesChanged`

Запрос на подписку изменений биометрических шаблонов для персоны. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Ответ:

```
{
```

```
}  
  "person_id": 149
```

- person_id – идентификатор изменённой персоны.

5.34. ReplicationBranchService

Файл replication_branch.proto предназначен для взаимодействия с модулем филиала репликации. Подробнее о репликации в документации «Бастион-3 – Репликация. Руководство администратора».

5.34.1. Коды типов сущностей репликации

Каждая сущность, участвующая в репликации, имеет свой тип, который идентифицируется числовым кодом. Коды типов сущностей указываются в изменениях репликации, в правилах разрешения конфликтов, в записях истории репликации и т.д.

Типы основных сущностей в репликации имеют следующие коды:

- словарное значение – 1;
- узел организационной структуры – 2;
- карта доступа – 3;
- персона – 4;
- фотография персоны – 5;
- биометрические шаблоны персоны – 6;
- персональный пропуск – 7;
- настройки репликации персонального пропуска – 8;
- филиал репликации – 9;
- глобальный уровень доступа персонального пропуска – 10;
- данные о внесении персоны в стоп-лист – 11;
- дескриптор дополнительных полей персональных пропусков – 12;
- значения дополнительных полей персонального пропуска – 13;
- справочник словарных значений – 14.

Типы сущностей транспортных пропусков имеют следующие коды:

- статус транспортного средства – 100;
- транспортное средство – 101;
- фотография транспортного средства – 102;
- транспортный пропуск – 103;
- дескриптор дополнительных полей транспортных пропусков – 104;
- значения дополнительных полей транспортного пропуска – 105.

5.34.2. Метод ApplyBranchChange

Запрос на применение исходящего изменения филиала. Запрос может быть выполнен только с использованием токена доступа. В качестве параметров запроса указываются идентификатор исходящего изменения и правила разрешения конфликтов.

Пример запроса:

```
{
  "branch_change_id": 7,
  "conflict_resolve_rules": {}
}
```

- `branch_change_id` – идентификатор исходящего изменения;
- `conflict_resolve_rules` – правила разрешения конфликтов.

При обработке изменения для разрешения конфликтов используются правила разрешения, указанные пользователем в `conflict_resolve_rules`, а также правила по умолчанию. В примере выше заданы пустые правила разрешения конфликтов, поэтому при обработке изменения будут использоваться только правила разрешения по умолчанию.

Тело запроса в примере выше эквивалентно следующему телу запроса:

```
{
  "branch_change_id": 7
}
```

Пример ответа на запрос в случае успешного применения изменения:

```
{
  "apply_result": {
    "is_applied": true,
    "conflict_info": null
  }
}
```

- `is_applied` – признак, указывающий на успешное применение изменения;
- `conflict_info` – информация о конфликте (если он возник).

Пример ответа на запрос в случае возникновения конфликта при применении изменения:

```
{
  "apply_result": {
    "is_applied": false,
    "conflict_info": {
      "field_compare_infos": [
        {
          "field_name": "Корпоративный код",
          "branch_field_value": "a6067ef0-5aac-4707-a063-6394a90d8dab",
          "center_field_value": "23d5a6f4-7b5f-4b26-85c9-1656b774cca4"
        }
      ]
    }
  }
}
```

```
    },
    {
      "field_name": "Дата создание личной карточки",
      "branch_field_value": "12.12.2025 15:51:46 +04:00",
      "center_field_value": "12.12.2025 14:50:55 +03:00"
    }
  ],
  "entity_additional_infos": [],
  "operation_description": "При добавлении персоны (Петров Пётр Петрович) возник конфликт дублирования данных.",
  "conflict_type": "CONFLICT_TYPE_DUPLICATE",
  "message": "Дублирование значений.",
  "entity_type_code": 4,
  "entity_id": 104,
  "entity_uid": "db0e59c7-e2e9-4fe4-b029-317985ea6e4a",
  "duplicate_uid": null,
  "remote_change_time": "2025-12-12T11:51:35.258Z",
  "remote_branch_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",
  "has_difference": true,
  "conflict_direction": "CONFLICT_DIRECTION_FROM_BRANCH_TO_CENTER"
}
}
}
```

В примере выше возник конфликт дубликатов в процессе передачи в центр персоны с идентификатором 104.

Если в ходе применения изменения возник конфликт, то поле `conflict_info` в ответе на запрос будет содержать информацию о конфликте:

- `operation_description` – описание конфликта;
- `conflict_type` – тип конфликта:
 - `CONFLICT_TYPE_VERSION` – конфликт версий;
 - `CONFLICT_TYPE_DUPLICATE` – конфликт, возникающий при нахождении дубликата сущности;
 - `CONFLICT_TYPE_LOGIC` – конфликт бизнес-логики;
- `message` – текстовые подробности конфликта. Может содержать описание причины конфликта;
- `entity_type_code` – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- `entity_id` – идентификатор сущности на филиале;
- `entity_uid` – глобальный идентификатор сущности;
- `remote_change_time` – время последнего изменения сущности на центре;
- `remote_branch_uid` – глобальный идентификатор филиала – источника последнего изменения сущности на центре;
- `has_difference` – признак наличия различий в локальном (на филиале) и удалённом (на центре) значениях сущности;
- `field_compare_infos` – описание различий в локальном (на филиале) и удалённом (на центре) значениях сущности;
- `entity_additional_infos` – дополнительное описание локального (на филиале) значения сущности;
- `conflict_direction` – направление изменения:
 - `CONFLICT_DIRECTION_FROM_BRANCH_TO_CENTER` – изменение, отправленное с филиала на центр;

- CONFLICT_DIRECTION_FROM_CENTER_TO_BRANCH – изменение, полученное филиалом с центра.

Информацию о конфликте (тип конфликта, код типа сущности, идентификатор сущности на филиале) следует использовать для указания правила разрешения конфликта при повторной попытке применения изменения. Правила разрешения конфликтов указываются в поле `conflict_resolve_rules` запроса на применение изменения.

Пример запроса на применение исходящего изменения с правилом "Объединить записи" для разрешения конфликта, приведённого в примере выше:

```
{
  "branch_change_id": 11,
  "conflict_resolve_rules": {
    "from_branch_to_center_rules": [
      {
        "conflict_direction": "CONFLICT_DIRECTION_FROM_BRANCH_TO_CENTER",
        "conflict_type": "CONFLICT_TYPE_DUPLICATE",
        "rule": "CONFLICT_RESOLVE_TYPE_MERGE_ENTITIES",
        "entity_type_code": 4,
        "entity_parameters": {
          "id": 104
        }
      }
    ]
  }
}
```

Параметры правила разрешения конфликта:

- `conflict_direction` – направление изменения;
- `conflict_type` – тип конфликта;
- `rule` – способ разрешения конфликта:
 - CONFLICT_RESOLVE_TYPE_APPLY_CENTER_VERSION – применить версию с центра репликации;
 - CONFLICT_RESOLVE_TYPE_OVERRIDE_CENTER_VERSION – переопределить значение в центре репликации;
 - CONFLICT_RESOLVE_TYPE_MERGE_ENTITIES – объединить записи;
 - CONFLICT_RESOLVE_TYPE_CREATE_DUPLICATE – создать дубликат записи;
- `entity_type_code` – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- `entity_parameters` – параметры для идентификации сущности:
 - `id` – идентификатор сущности на филиале.

Отменить применение исходящего изменения можно с помощью запроса на снятие актуальности с изменения (см. раздел «[Метод AvoidBranchChange](#)»).

5.34.3. Метод ApplyCenterChange

Запрос на применение входящего изменения, полученного с центра репликации. Запрос может быть выполнен только с использованием токена доступа. В качестве параметров запроса указываются идентификатор входящего изменения и правила разрешения конфликтов.

Пример запроса:

```
{
  "center_change_id": "15",
  "conflict_resolve_rules": {}
}
```

- center_change_id – идентификатор входящего изменения;
- conflict_resolve_rules – правила разрешения конфликтов.

При обработке изменения для разрешения конфликтов используются правила разрешения, указанные пользователем в conflict_resolve_rules, а также правила по умолчанию. В примере выше заданы пустые правила разрешения конфликтов, поэтому при обработке изменения будут использоваться только правила разрешения по умолчанию.

Тело запроса в примере выше эквивалентно следующему телу запроса:

```
{
  "center_change_id": "15"
}
```

Пример ответа на запрос в случае успешного применения изменения:

```
{
  "apply_result": {
    "is_applied": true,
    "conflict_info": null
  }
}
```

- is_applied – признак, указывающий на успешное применение изменения;
- conflict_info – информация о конфликте (если он возник).

Пример ответа на запрос в случае возникновения конфликта при применении изменения:

```
{
  "apply_result": {
    "is_applied": false,
    "conflict_info": {
      "field_compare_infos": [
        {
          "field_name": "Корпоративный код",
          "branch_field_value": "6b1ecf5d-cfa9-447c-9cc1-97b04ed13c86",
          "center_field_value": "0336e6a3-b043-42e2-b3f3-1115ae4c92b4"
        },
        {
          "field_name": "Дата создание личной карточки",
          "branch_field_value": "12.12.2025 17:01:07 +04:00",
          "center_field_value": "12.12.2025 16:01:39 +03:00"
        }
      ]
    }
  }
}
```

```

        "entity_additional_infos": [],
        "operation_description": "При добавлении персоны (Сидоров Иван Петрович) возник
конфликт дублирования данных.",
        "conflict_type": "CONFLICT_TYPE_DUPLICATE",
        "message": "Обнаружена локальная дублирующая запись.",
        "entity_type_code": 4,
        "entity_id": 105,
        "entity_uid": "21a898d9-4db4-4c13-9f12-d7b6f057a9c3",
        "duplicate_uid": null,
        "remote_change_time": "2025-12-12T13:02:02.221Z",
        "remote_branch_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",
        "has_difference": true,
        "conflict_direction": "CONFLICT_DIRECTION_FROM_CENTER_TO_BRANCH"
    }
}
}

```

В примере выше при добавлении в филиале персоны (у которой глобальный идентификатор 21a898d9-4db4-4c13-9f12-d7b6f057a9c3), полученной с центра, возник конфликт дубликатов: на филиале найдена дублирующая персона (у которой на филиале идентификатор равен 105).

Если в ходе применения изменения возник конфликт, то поле `conflict_info` в ответе на запрос будет содержать информацию о конфликте. Описание параметров конфликта представлено в разделе «Метод `ApplyBranchChange`».

Информацию о конфликте (тип конфликта, код типа сущности, глобальный идентификатор сущности) следует использовать для указания правила разрешения конфликта при повторной попытке применения изменения. Правила разрешения конфликтов указываются в поле `conflict_resolve_rules` запроса на применение изменения.

Пример запроса на применение входящего изменения с правилом "Объединить записи" для разрешения конфликта, приведённого в примере выше:

```

{
  "center_change_id": "24",
  "conflict_resolve_rules": {
    "from_center_to_branch_rules": [
      {
        "conflict_direction": "CONFLICT_DIRECTION_FROM_CENTER_TO_BRANCH",
        "conflict_type": "CONFLICT_TYPE_DUPLICATE",
        "rule": "CONFLICT_RESOLVE_TYPE_MERGE_ENTITIES",
        "entity_type_code": 4,
        "entity_parameters": {
          "uid": "21a898d9-4db4-4c13-9f12-d7b6f057a9c3"
        }
      }
    ]
  }
}

```

Параметры правила разрешения конфликта:

- `conflict_direction` – направление изменения;
- `conflict_type` – тип конфликта;
- `rule` – способ разрешения конфликта:

- CONFLICT_RESOLVE_TYPE_APPLY_CENTER_VERSION – применить версию с центра репликации;
- CONFLICT_RESOLVE_TYPE_OVERRIDE_CENTER_VERSION – переопределить значение в центре репликации;
- CONFLICT_RESOLVE_TYPE_MERGE_ENTITIES – объединить записи;
- CONFLICT_RESOLVE_TYPE_CREATE_DUPLICATE – создать дубликат записи;
- entity_type_code – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- entity_parameters – параметры для идентификации сущности:
 - uid – глобальный идентификатор сущности.

Отменить применение входящего изменения можно с помощью запроса на снятие актуальности с изменения (см. раздел «[Метод AvoidCenterChange](#)»).

5.34.4. Метод AvoidBranchChange

Запрос на снятие актуальности с исходящего изменения: изменение будет помечено как применённое с комментарием "Применение отклонено оператором.". Запрос может быть выполнен только с использованием токена доступа. В качестве параметра запроса указывается идентификатор исходящего изменения. Ответ на запрос – сообщение с пустым телом. Подробнее о мониторинге и управлению репликацией написано в разделе «Мониторинг репликации» документа «Бастион-3 – Репликация. Руководство администратора».

Пример запроса:

```
{
  "branch_change_id": 3
}
```

- branch_change_id – идентификатор исходящего изменения.

5.34.5. Метод AvoidCenterChange

Запрос на снятие актуальности со входящего изменения: изменение будет помечено как применённое с комментарием "Применение отклонено оператором.". Запрос может быть выполнен только с использованием токена доступа. В качестве параметра запроса указывается идентификатор входящего изменения. Ответ на запрос – сообщение с пустым телом. Подробнее о мониторинге и управлению репликацией написано в разделе «Мониторинг репликации» документа «Бастион-3 – Репликация. Руководство администратора».

Пример запроса:

```
{
  "center_change_id": "8"
}
```

Параметры запроса:

- center_change_id – идентификатор входящего изменения.

5.34.6. Метод BranchApplyMessageReceived

Запрос на получение сообщений логов, генерируемых филиалом при обработке изменений репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа:

```
{
  "apply_message": {
    "date_time": "2025-12-09T12:58:58.790Z",
    "kind": "APPLY_MESSAGE_KIND_NORMAL",
    "message": "Начало процесса получения новых изменений с центра репликации.",
    "entity_type_code": null,
    "entity_id": null,
    "entity_uid": null
  }
}
```

Ответ содержит в себе следующие параметры сообщения:

- `date_time` – дата и время создания сообщения;
- `kind` – тип сообщения:
 - `APPLY_MESSAGE_KIND_DEBUG` – отладочное сообщение;
 - `APPLY_MESSAGE_KIND_NORMAL` – информационное сообщение;
 - `APPLY_MESSAGE_KIND_ERROR` – сообщение об ошибке;
- `message` – текст сообщения;
- `entity_type_code` – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- `entity_id` – идентификатор сущности на филиале;
- `entity_uid` – глобальный идентификатор сущности.

5.34.7. Метод BranchChangeStateChanged

Запрос на отслеживание изменения состояний исходящих изменений филиала. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "branch_change_info": {
    "id": 5,
    "entity_type_code": 9,
    "change_type": "CHANGE_TYPE_ADD",
    "state": "BRANCH_CHANGE_STATE_COMPLETED",
    "entity_id": 104,
    "create_time": "2025-12-09T14:36:39.911Z",
    "state_change_time": "2025-12-11T08:08:22.619Z",
    "state_comments": "",
    "entity_uid": "61ec7211-8ec8-4edf-bc74-fdbcbcd446b6f"
  }
}
```

Ответ на запрос содержит актуальные параметры изменения, включая его состояние. Описание параметров представлено в разделе «[Метод GetBranchChanges](#)».

5.34.8. Метод BranchListChanged

Запрос на получение уведомлений об изменении списка филиалов репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. При наступлении события поступает ответ с пустым телом.

5.34.9. Метод CardOwnerChanged

Запрос на отслеживание события об изменении филиала – владельца карты доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа при наступлении события:

```
{
  "card_info": {
    "card_id": 301,
    "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
  }
}
```

Ответное сообщение содержит следующие параметры события:

- card_id – идентификатор карты доступа на филиале;
- owner_uid – глобальный идентификатор нового владельца карты доступа – участника репликации.

5.34.10. Метод CenterChangeStateChanged

Запрос на отслеживание изменения состояний входящих изменений филиала. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "center_change_info": {
    "id": "14",
    "source_branch_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",
    "source_branch_time": "2025-12-09T14:19:32.098Z",
    "center_time": "2025-12-09T14:19:32.098Z",
    "state_change_time": "2025-12-09T14:19:40.728Z",
    "entity_type_code": 10,
    "change_type": "CHANGE_TYPE_UPDATE",
    "state": "CENTER_CHANGE_STATE_APPLIED",
    "state_comments": "",
    "entity_uid": "f17baf56-e3dd-4955-b25a-d7b56da859fe"
  }
}
```

Ответ на запрос содержит актуальные параметры изменения, включая его состояние. Описание параметров представлено в разделе «Метод [GetCenterChanges](#)».

5.34.11. Метод [ClearReplicationProtocol](#)

Запрос на очистку протокола репликации на филиале. Запрос может быть выполнен только с использованием токена доступа. В качестве параметра запроса указывается период, за который нужно оставить историю репликации. Ответ на запрос – сообщение с пустым телом. Подробнее об очистке протокола репликации написано в разделе «Очистка протокола репликации» документа «Бастион-3 – Репликация. Руководство администратора».

Пример запроса:

```
{
  "keep_history_depth": "KEEP_HISTORY_DEPTH_ONE_MONTH"
}
```

- keep_history_depth – период, за который нужно оставить историю репликации:
 - KEEP_HISTORY_DEPTH_ZERO – очищать всю историю репликации;
 - KEEP_HISTORY_DEPTH_ONE_WEEK – оставлять историю репликации за последнюю неделю;
 - KEEP_HISTORY_DEPTH_ONE_MONTH – оставлять историю репликации за последний месяц;
 - KEEP_HISTORY_DEPTH_TWO_MONTH – оставлять историю репликации за последние два месяца;
 - KEEP_HISTORY_DEPTH_ONE_QUARTER – оставлять историю репликации за последние три месяца;
 - KEEP_HISTORY_DEPTH_ALL – не очищать историю репликации.

5.34.12. Метод [GetAllCardsReplicationInfo](#)

Запрос метаданных репликации всех карт доступа, которые имеются на филиале и участвуют в репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "card_replication_infos": [
    {
      "card_id": 101,
      "owner_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a"
    },
    {
      "card_id": 121,
      "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
    }
  ]
}
```

- card_id – идентификатор карты доступа;

- owner_uid – глобальный идентификатор филиала – владельца карты доступа.

5.34.13. Метод GetAllOrganizationsReplicationInfo

Запрос метаданных репликации всех организаций/подразделений, которые имеются на филиале и участвуют в репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "organization_node_replication_infos": [
    {
      "organization_node_id": 0,
      "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
    },
    {
      "organization_node_id": 100,
      "owner_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a"
    },
    {
      "organization_node_id": 101,
      "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
    }
  ]
}
```

- organization_node_id – идентификатор организации/подразделения;
- owner_uid – глобальный идентификатор филиала – владельца организации/подразделения.

5.34.14. Метод GetAllPassCategoryOrganizationReplicationProperties

Запрос всех настроек репликации по умолчанию для персональных пропусков. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. Подробнее о настройках репликации по умолчанию написано в разделе «Настройка параметров репликации пропусков по умолчанию для подразделений и категорий пропусков» документа «Бастион-3 – Репликация. Руководство администратора».

Пример ответа на запрос:

```
{
  "replication_properties": [
    {
      "target_branch_ids": [
        100,
        101
      ],
      "target_branch_custom_global_access_level_map": {
        "100": 2
      },
      "pass_category_id": 2,
      "organization_node_id": 101,
      "global_access_level_id": 1
    }
  ],
}
```

```

    {
      "target_branch_ids": [
        101
      ],
      "target_branch_custom_global_access_level_map": {},
      "pass_category_id": 2,
      "organization_node_id": 102,
      "global_access_level_id": 3
    }
  ]
}

```

- `pass_category_id` – идентификатор категории пропусков;
- `organization_node_id` – идентификатор организации/подразделения;
- `global_access_level_id` – идентификатор глобального уровня доступа;
- `target_branch_ids` – идентификаторы филиалов – пунктов назначения;
- `target_branch_custom_global_access_level_map` – словарь отдельных назначений глобальных уровней доступа для филиалов – пунктов назначения. В филиале, идентификатор которого указан в качестве ключа, для пропусков будет действовать глобальный уровень доступа, идентификатор которого указан в качестве значения.

5.34.15. Метод `GetAllPassesReplicationInfo`

Запрос метаданных репликации всех персональных пропусков, которые имеются на филиале и участвуют в репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```

{
  "pass_replication_infos": [
    {
      "pass_id": 101,
      "owner_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a"
    },
    {
      "pass_id": 102,
      "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
    }
  ]
}

```

- `pass_id` – идентификатор пропуска;
- `owner_uid` – глобальный идентификатор филиала – владельца пропуска.

5.34.16. Метод `GetAllPersonsReplicationInfo`

Запрос метаданных репликации всех персон, которые имеются на филиале и участвуют в репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "person_replication_infos": [
    {
      "person_id": 101,
      "owner_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a"
    },
    {
      "person_id": 102,
      "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
    }
  ]
}
```

- person_id – идентификатор персоны;
- owner_uid – глобальный идентификатор филиала – владельца персоны.

5.34.17. Метод GetBranchChangeApplyInfo

Запрос данных о применении исходящего изменения на филиалах. Запрос может быть выполнен только с использованием токена доступа. В качестве параметра запроса указывается идентификатор исходящего изменения.

Пример запроса:

```
{
  "branch_change_id": 10
}
```

- branch_change_id – идентификатор исходящего изменения.

Пример ответа на запрос:

```
{
  "change_apply_infos": [
    {
      "center_change_id": "18",
      "branch_uid": "8f4fac81-2662-4521-83d1-cc8761f5dabe",
      "branch_name": "Филиал 2",
      "status": "CHANGE_STATUS_ON_BRANCH_NEW",
      "apply_time": "2025-12-10T05:30:56.405Z",
      "apply_result": null
    },
    {
      "center_change_id": "18",
      "branch_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",
      "branch_name": "Центр",
      "status": "CHANGE_STATUS_ON_BRANCH_APPLIED",
      "apply_time": "2025-12-10T05:57:28.710Z",
      "apply_result": null
    }
  ]
}
```

```
}
```

Ответ содержит записи с информацией о применении изменения на каждом из филиалов – пунктов назначения изменения. Каждая запись содержит следующую информацию:

- `center_change_id` – идентификатор изменения на центре;
- `branch_uid` – глобальный идентификатор целевого филиала;
- `branch_name` – имя филиала;
- `status` – статус применения изменения на филиале:
 - `CHANGE_STATUS_ON_BRANCH_NEW` – новое изменение;
 - `CHANGE_STATUS_ON_BRANCH_RECEIVED` – изменение получено филиалом;
 - `CHANGE_STATUS_ON_BRANCH_ERROR_ON_APPLY` – при применении филиалом изменения произошла ошибка;
 - `CHANGE_STATUS_ON_BRANCH_CONFLICT` – при применении филиалом изменения возник конфликт;
 - `CHANGE_STATUS_ON_BRANCH_APPLIED` – изменение успешно применено филиалом;
- `apply_time` – время обновления статуса изменения на филиале;
- `apply_result` – результат применения изменения.

5.34.18. Метод `GetBranchChanges`

Запрос исходящих изменений филиала репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "branch_changes": [
    {
      "id": 1,
      "entity_type_code": 10,
      "change_type": "CHANGE_TYPE_ADD",
      "state": "BRANCH_CHANGE_STATE_CREATED",
      "entity_id": 1,
      "create_time": {
        "seconds": "1765198349",
        "nanos": 412097000
      },
      "state_change_time": {
        "seconds": "1765198349",
        "nanos": 412121000
      },
      "state_comments": null,
      "entity_uid": null
    },
    {
      "id": 2,
      "entity_type_code": 10,
      "change_type": "CHANGE_TYPE_ADD",
      "state": "BRANCH_CHANGE_STATE_CREATED",
      "entity_id": 2,
      "create_time": {
```

```

        "seconds": "1765198349",
        "nanos": 434951000
    },
    "state_change_time": {
        "seconds": "1765198349",
        "nanos": 434952000
    },
    "state_comments": null,
    "entity_uid": null
}
]
}

```

Ответное сообщение содержит список исходящих изменений, имеющихся на филиале. О каждом изменении представлена следующая информация:

- id – идентификатор изменения на филиале;
- entity_type_code – код типа изменённой сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- change_type – тип операции над сущностью:
 - CHANGE_TYPE_ADD – добавление сущности;
 - CHANGE_TYPE_UPDATE – обновление сущности;
 - CHANGE_TYPE_DELETE – удаление сущности;
 - CHANGE_TYPE_MERGE – объединение сущности;
- state – статус изменения:
 - BRANCH_CHANGE_STATE_CREATED – создано;
 - BRANCH_CHANGE_STATE_SENT_TO_CENTER – в процессе передачи в центр;
 - BRANCH_CHANGE_STATE_ERROR_ON_APPLY – ошибка при передаче в центр;
 - BRANCH_CHANGE_STATE_NESTED_CONFLICT – конфликт применения;
 - BRANCH_CHANGE_STATE_CONFLICTED – конфликт применения;
 - BRANCH_CHANGE_STATE_COMPLETED – завершено;
- entity_id – идентификатор сущности на филиале;
- create_time – дата и время создания изменения на филиале;
- state_change_time – дата и время обновления статуса;
- state_comments – комментарий к статусу;
- entity_uid – глобальный идентификатор сущности.

5.34.19. Метод GetBranches

Запрос списка филиалов репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```

{
  "branches": [
    {
      "id": 0,

```

```

        "uid": {
            "value": "be628dee-85f5-45cf-a7f2-e3b5845e6a59"
        },
        "name": "Центр",
        "is_pass_target": false,
        "is_initialized": true
    },
    {
        "id": 101,
        "uid": null,
        "name": "Филиал 2",
        "is_pass_target": true,
        "is_initialized": false
    },
    {
        "id": 100,
        "uid": null,
        "name": "Филиал",
        "is_pass_target": true,
        "is_initialized": false
    }
]
}

```

Ответное сообщение содержит список всех филиалов репликации. О каждом филиале представлена следующая информация:

- `id` – идентификатор филиала на филиале репликации. Филиал, которому отправлен запрос, в качестве идентификатора будет иметь значение 0;
- `uid` – глобальный идентификатор филиала;
- `name` – имя филиала;
- `is_pass_target` – признак, указывающий, включать ли филиал в список доступных для выбора пунктов назначения персонального пропуска;
- `is_initialized` – признак, указывающий, связан ли филиал с реальным объектом ПК «Бастион-3».

5.34.20. Метод `GetBranchFilteredHistory`

Запрос истории репликации филиала. Запрос может быть выполнен только с использованием токена доступа. Возвращает записи истории в обратном хронологическом порядке: от поздних к ранним. Поддерживает "постраничную" загрузку истории с указанием идентификатора первой (самой поздней) записи страницы и количества записей для загрузки. Есть возможность запрашивать историю как для одной сущности, так и для всех. Для более точного поиска есть возможность задания фильтров для полей записей истории. Подробнее про просмотр истории репликации написано в разделе «Просмотр истории репликации» документа «Бастион-3 – Репликация. Руководство администратора».

Пример запроса записей истории репликации филиала для всех сущностей:

1. фильтры записей: описание сущности должно включать в себя подстроку "Иванов Иван", а изменение репликации, при обработке которого сущность была изменена, должно быть исходящим;
2. параметры страницы истории: загрузить первую страницу (для этого в качестве идентификатора первой (самой поздней) записи страницы нужно указать любое отрицательное значение, в данном случае -1), количество записей для загрузки – 10.

```
{
  "filter": {
    "all": {},
    "fields_filter": {
      "change_direction_rule": {
        "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER"
      },
      "entity_description_rule": {
        "entity_description": "Иванов Иван"
      }
    }
  },
  "last_history_entry_id": -1,
  "packet_size": 10
}
```

Параметры запроса истории включают в себя:

- **fields_filter** – описание фильтров для полей записей истории. Доступные фильтры:
 - **change_time_rule** – фильтр по датам записей. Записи истории должны попадать в заданный период времени. Включает в себя следующие параметры: **start_date** – нижняя граница периода, **end_date** – верхняя граница периода;
 - **change_direction_rule** – фильтр по направлению изменения;
 - **entity_type_rule** – фильтр по типу сущности. Параметр фильтра: **entity_type_code** – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
 - **entity_uid_rule** – фильтр по глобальному идентификатору сущности;
 - **entity_description_rule** – фильтр по описанию сущности;
 - **change_source_rule** – фильтр по имени филиала – источника изменения;
- **last_history_entry_id** – идентификатор первой (самой поздней) записи страницы истории. Для запроса первой страницы истории следует указывать любое отрицательное значение (например, -1);
- **packet_size** – количество записей страницы для загрузки.

Пример ответа на запрос выше:

```
{
  "history_entries": [
    {
      "id": "43",
      "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER",
      "change_type": "CHANGE_TYPE_UPDATE",
      "entity_type_code": 8,
      "entity_uid": "c0bdd3ad-e1a5-4c80-a657-cc6f86b61d4b",
      "entity_description": "Иванов Иван Иванович карта: 00000064C7CB",
      "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
      "change_source_name": "Филиал",
      "change_time": "2025-12-11T05:30:56.353Z",
      "comment": null
    },
    {
      "id": "42",
      "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER",

```

```

        "change_type": "CHANGE_TYPE_UPDATE",
        "entity_type_code": 5,
        "entity_uid": "b9b9ba23-8676-4b43-aaf6-372d00f91a96",
        "entity_description": "Иванов Иван Иванович",
        "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
        "change_source_name": "Филиал",
        "change_time": "2025-12-10T19:49:28.429Z",
        "comment": null
    },
    {
        "id": "41",
        "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER",
        "change_type": "CHANGE_TYPE_UPDATE",
        "entity_type_code": 8,
        "entity_uid": "c0bdd3ad-e1a5-4c80-a657-cc6f86b61d4b",
        "entity_description": "Иванов Иван Иванович карта: 00000064C7CB",
        "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
        "change_source_name": "Филиал",
        "change_time": "2025-12-10T19:48:53.456Z",
        "comment": null
    },
    ...
]
}

```

Поля записи истории включают в себя:

- id – идентификатор записи;
- change_direction – направление изменения: входящее/исходящее;
- change_type – тип операции над сущностью;
- entity_type_code – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- entity_uid – глобальный идентификатор сущности;
- entity_description – описание сущности;
- change_source_uid – глобальный идентификатор филиала – источника изменения;
- change_source_name – имя филиала – источника изменения;
- change_time – дата и время изменения;
- comment – комментарий.

Пример запроса записей истории репликации филиала для одной сущности по коду её типа и идентификатору на филиале:

1. условие загрузки: история организации с идентификатором 105;
2. фильтр записей: источник изменения – филиал, имя которого содержит подстроку "Филиал";
3. параметры страницы истории: загрузить самую первую страницу, количество записей в странице – 10.

```

{
  "filter": {
    "by_entity_id": {
      "entity_id": 105,
      "entity_type_code": 2
    },
    "fields_filter": {

```

```

        "change_source_rule": {
            "change_source_name": "Филиал"
        }
    },
    "last_history_entry_id": -1,
    "packet_size": 10
}

```

Параметры загрузки истории одной сущности по её идентификатору на филиале:

- entity_id – идентификатор сущности на филиале;
- entity_type_code – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»).

Пример ответа на запрос выше:

```

{
  "history_entries": [
    {
      "id": "35",
      "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER",
      "change_type": "CHANGE_TYPE_ADD",
      "entity_type_code": 2,
      "entity_uid": "6e11ee32-4e15-46fa-bb58-2101028225da",
      "entity_description": "Организация: Организация 1234",
      "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
      "change_source_name": "Филиал",
      "change_time": "2025-12-10T19:48:53.145Z",
      "comment": null
    }
  ]
}

```

Пример запроса записей истории репликации филиала для одной сущности по её глобальному идентификатору:

1. условие загрузки: история сущности с идентификатором "6e11ee32-4e15-46fa-bb58-2101028225da";
2. фильтр записей: источник изменения – филиал, имя которого содержит подстроку "Центр";
3. параметры страницы истории: загрузить самую первую страницу, количество записей в странице – 10.

```

{
  "filter": {
    "by_entity_uid": {
      "entity_uid": "6e11ee32-4e15-46fa-bb58-2101028225da"
    },
    "fields_filter": {
      "change_source_rule": {
        "change_source_name": "Центр"
      }
    }
  },
  "last_history_entry_id": -1,
}

```

```
    "packet_size": 10
  }
```

Параметры загрузки истории одной сущности по её глобальному идентификатору:

- `entity_uid` – глобальный идентификатор сущности.

Пример ответа на запрос выше:

```
{
  "history_entries": [
    {
      "id": "45",
      "change_direction": "CHANGE_DIRECTION_FROM_CENTER_TO_BRANCH",
      "change_type": "CHANGE_TYPE_UPDATE",
      "entity_type_code": 2,
      "entity_uid": "6e11ee32-4e15-46fa-bb58-2101028225da",
      "entity_description": "Организация: Организация 1234_22",
      "change_source_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",
      "change_source_name": "Центр",
      "change_time": "2025-12-11T13:48:29.309Z",
      "comment": null
    }
  ]
}
```

5.34.21. Метод `GetBranchFilteredHistoryDateRange`

Запрос периода времени истории репликации филиала, удовлетворяющей заданным фильтрам. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "filter": {
    "all": {},
    "fields_filter": {
      "change_direction_rule": {
        "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER"
      },
      "entity_description_rule": {
        "entity_description": "Иванов Иван"
      }
    }
  }
}
```

Описание фильтров приведено в разделе «[Метод `GetBranchFilteredHistory`](#)».

Пример ответа на запрос:

```
{
  "start_date": "2025-12-10T19:48:53.304Z",
```

```

    "end_date": "2025-12-10T19:49:28.429Z"
  }

```

- start_date – нижняя граница периода;
- end_date – верхняя граница периода.

5.34.22. Метод GetBranchFilteredHistorySize

Запрос количества записей истории репликации филиала, удовлетворяющих заданным фильтрам. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "filter": {
    "all": {},
    "fields_filter": {
      "change_direction_rule": {
        "change_direction": "CHANGE_DIRECTION_FROM_BRANCH_TO_CENTER"
      },
      "entity_description_rule": {
        "entity_description": "Иванов Иван"
      }
    }
  }
}

```

Описание фильтров приведено в разделе «[Метод GetBranchFilteredHistory](#)».

Пример ответа на запрос:

```

{
  "history_size": "7"
}

```

- history_size – количество записей истории репликации филиала, удовлетворяющих заданным фильтрам.

5.34.23. Метод GetCenterChanges

Запрос входящих изменений филиала репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```

{
  "center_changes": [
    {
      "id": "3",
      "source_branch_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",

```

```

        "source_branch_time": "2025-12-09T08:15:52.440Z",
        "center_time": "2025-12-09T08:15:52.440Z",
        "state_change_time": "2025-12-09T08:19:28.664Z",
        "entity_type_code": 10,
        "change_type": "CHANGE_TYPE_UPDATE",
        "state": "CENTER_CHANGE_STATE_APPLIED",
        "state_comments": "",
        "entity_uid": "2aeb02af-c16a-4368-8450-e33fa89849fb"
    },
    {
        "id": "4",
        "source_branch_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a",
        "source_branch_time": "2025-12-09T08:15:52.592Z",
        "center_time": "2025-12-09T08:15:52.592Z",
        "state_change_time": "2025-12-09T08:19:28.723Z",
        "entity_type_code": 10,
        "change_type": "CHANGE_TYPE_UPDATE",
        "state": "CENTER_CHANGE_STATE_APPLIED",
        "state_comments": "",
        "entity_uid": "87e23c8f-6d68-471c-a624-abd367743286"
    }
]
}

```

Ответное сообщение содержит список входящих изменений, имеющих на филиале. О каждом изменении представлена следующая информация:

- id – идентификатор изменения;
- source_branch_uid – глобальный идентификатор филиала – источника изменения;
- source_branch_time – дата и время создания изменения на филиале-источнике;
- center_time – дата и время принятия изменения центром;
- state_change_time – дата и время обновления статуса;
- entity_type_code – код типа изменённой сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- change_type – тип операции над сущностью:
 - CHANGE_TYPE_ADD – добавление сущности;
 - CHANGE_TYPE_UPDATE – обновление сущности;
 - CHANGE_TYPE_DELETE – удаление сущности;
 - CHANGE_TYPE_MERGE – объединение сущности;
- state – статус изменения:
 - CENTER_CHANGE_STATE_RECEIVED – ожидает применения;
 - CENTER_CHANGE_STATE_ERROR_ON_APPLY – ошибка применения;
 - CENTER_CHANGE_STATE_CONFLICT – конфликт применения;
 - CENTER_CHANGE_STATE_APPLIED – применено;
- state_comments – комментарий к статусу;
- entity_uid – глобальный идентификатор сущности.

5.34.24. Метод GetCenterFilteredHistory

Запрос истории репликации центра. Запрос может быть выполнен только с использованием токена доступа. Возвращает записи истории в обратном хронологическом порядке: от поздних к ранним. Поддерживает "постраничную" загрузку истории с указанием идентификатора первой (самой поздней) записи страницы и количества записей для загрузки. Есть возможность запрашивать историю как для одной сущности, так и для всех. Для более точного поиска есть возможность задания фильтров для полей записей истории. Подробнее про просмотр истории репликации написано в разделе «Просмотр истории репликации» документа «Бастион-3 – Репликация. Руководство администратора».

Пример запроса записей истории репликации центра для всех сущностей:

1. фильтры записей: загрузить историю изменения карт доступа филиалом, имя которого содержит подстроку "Филиал";
2. параметры страницы истории: загрузить первую страницу (для этого в качестве идентификатора первой (самой поздней) записи страницы нужно указать любое отрицательное значение, в данном случае -1), количество записей для загрузки – 10.

```
{
  "filter": {
    "all": {},
    "fields_filter": {
      "change_source_rule": {
        "change_source_name": "Филиал"
      },
      "entity_type_rule": {
        "entity_type_code": 3
      }
    }
  },
  "last_history_entry_id": -1,
  "packet_size": 10
}
```

Параметры запроса истории включают в себя:

- `fields_filter` – описание фильтров для полей записей истории. Доступные фильтры:
 - `change_time_rule` – фильтр по датам записей. Записи истории должны попадать в заданный период времени. Включает в себя следующие параметры: `start_date` – нижняя граница периода, `end_date` – верхняя граница периода;
 - `entity_type_rule` – фильтр по типу сущности. Параметр фильтра: `entity_type_code` – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
 - `entity_uid_rule` – фильтр по глобальному идентификатору сущности;
 - `entity_description_rule` – фильтр по описанию сущности;
 - `change_source_rule` – фильтр по имени филиала – источника изменения;
- `last_history_entry_id` – идентификатор первой (самой поздней) записи страницы истории. Для запроса первой страницы истории следует указывать любое отрицательное значение (например, -1);
- `packet_size` – количество записей страницы для загрузки.

Пример ответа на запрос выше:

```
{
  "history_entries": [
    {
      "id": "38",
      "change_type": "CHANGE_TYPE_ADD",
      "entity_type_code": 3,
      "entity_uid": "c4ad09a7-653e-42ae-b85b-321e353a6361",
      "entity_description": "00000064C7CB",
      "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
      "change_source_name": "Филиал",
      "change_time": "2025-12-10T19:48:52.798Z",
      "comment": null
    },
    {
      "id": "25",
      "change_type": "CHANGE_TYPE_ADD",
      "entity_type_code": 3,
      "entity_uid": "ce64de92-4795-4519-834b-aba1820b291d",
      "entity_description": "000000662E6A",
      "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
      "change_source_name": "Филиал",
      "change_time": "2025-12-09T08:52:31.923Z",
      "comment": null
    }
  ]
}
```

Поля записи истории включают в себя:

- id – идентификатор записи;
- change_type – тип операции над сущностью;
- entity_type_code – код типа сущности (см. раздел «[Коды типов сущностей репликации](#)»);
- entity_uid – глобальный идентификатор сущности;
- entity_description – описание сущности;
- change_source_uid – глобальный идентификатор филиала – источника изменения;
- change_source_name – имя филиала – источника изменения;
- change_time – дата и время изменения;
- comment – комментарий.

Пример запроса записей истории репликации центра для одной сущности по её глобальному идентификатору:

1. условие загрузки: история сущности с идентификатором "c0bdd3ad-e1a5-4c80-a657-cc6f86b61d4b";
2. фильтры записей не заданы;
3. параметры страницы истории: загрузить самую первую страницу, количество записей в странице – 10.

```
{
  "filter": {
    "by_entity_uid": {
      "entity_uid": "c0bdd3ad-e1a5-4c80-a657-cc6f86b61d4b"
    }
  }
}
```

```

    }
  },
  "last_history_entry_id": -1,
  "packet_size": 10
}

```

Параметры загрузки истории одной сущности по её глобальному идентификатору:

- `entity_uid` – глобальный идентификатор сущности.

Пример ответа на запрос выше:

```

{
  "history_entries": [
    {
      "id": "45",
      "change_type": "CHANGE_TYPE_UPDATE",
      "entity_type_code": 8,
      "entity_uid": "c0bdd3ad-e1a5-4c80-a657-cc6f86b61d4b",
      "entity_description": "Иванов Иван Иванович, карта доступа: 00000064C7CB",
      "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
      "change_source_name": "Филиал",
      "change_time": "2025-12-11T05:30:56.286Z",
      "comment": null
    },
    {
      "id": "42",
      "change_type": "CHANGE_TYPE_ADD",
      "entity_type_code": 7,
      "entity_uid": "c0bdd3ad-e1a5-4c80-a657-cc6f86b61d4b",
      "entity_description": "Иванов Иван Иванович, карта доступа: 00000064C7CB",
      "change_source_uid": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
      "change_source_name": "Филиал",
      "change_time": "2025-12-10T19:48:53.308Z",
      "comment": null
    }
  ]
}

```

5.34.25. Метод `GetCenterFilteredHistoryDateRange`

Запрос периода времени истории репликации центра, удовлетворяющей заданным фильтрам. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "filter": {
    "all": {},
    "fields_filter": {
      "entity_type_rule": {
        "entity_type_code": 3
      },
      "change_source_rule": {
        "change_source_name": "Филиал"
      }
    }
  }
}

```

```

    }
  }
}

```

Описание фильтров приведено в разделе «[Метод GetCenterFilteredHistory](#)».

Пример ответа на запрос:

```

{
  "start_date": "2025-12-09T08:52:31.923Z",
  "end_date": "2025-12-10T19:48:52.798Z"
}

```

- start_date – нижняя граница периода;
- end_date – верхняя граница периода.

5.34.26. Метод GetCenterFilteredHistorySize

Запрос количества записей истории репликации центра, удовлетворяющих заданным фильтрам. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```

{
  "filter": {
    "all": {},
    "fields_filter": {
      "entity_type_rule": {
        "entity_type_code": 3
      },
      "change_source_rule": {
        "change_source_name": "Филиал"
      }
    }
  }
}

```

Описание фильтров приведено в разделе «[Метод GetCenterFilteredHistory](#)».

Пример ответа на запрос:

```

{
  "history_size": "2"
}

```

- history_size – количество записей истории репликации центра, удовлетворяющих заданным фильтрам.

5.34.27. Метод `GetClearReplicationProtocolScheduleSettings`

Запрос параметров расписания очистки протокола репликации на филиале. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. Подробнее о расписании очистки протокола репликации написано в разделе «Настройка расписаний» документа «Бастион-3 – Репликация. Руководство администратора».

Пример ответа на запрос:

```
{
  "settings": {
    "is_enabled": true,
    "frequency": "CLEAR_REPLICATION_PROTOCOL_SCHEDULE_FREQUENCY_MONTHLY",
    "start_day": "CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_LAST_DAY_OF_THE_MONTH",
    "start_time": {
      "hours": 7,
      "minutes": 5,
      "seconds": 0,
      "nanos": 0
    },
    "keep_history_depth": "KEEP_HISTORY_DEPTH_ONE_WEEK"
  }
}
```

Ответ на запрос включает в себя следующие параметры:

- `is_enabled` – признак, указывающий, включено ли выполнение очистки протокола репликации по расписанию;
- `frequency` – частота выполнения очистки протокола репликации:
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_FREQUENCY_WEEKLY` – раз в неделю;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_FREQUENCY_MONTHLY` – раз в месяц;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_FREQUENCY_BIMONTHLY` – раз в два месяца;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_FREQUENCY_QUARTERLY` – раз в три месяца;
- `start_day` – день выполнения:
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_MONDAY` – понедельник;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_TUESDAY` – вторник;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_WEDNESDAY` – среда;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_THURSDAY` – четверг;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_FRIDAY` – пятница;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_SATURDAY` – суббота;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_SUNDAY` – воскресенье;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_FIRST_DAY_OF_THE_MONTH` – первый день месяца;
 - `CLEAR_REPLICATION_PROTOCOL_SCHEDULE_DAY_LAST_DAY_OF_THE_MONTH` – последний день месяца;
- `start_time` – время выполнения очистки протокола репликации по расписанию в UTC;
- `keep_history_depth` – интервал времени, за который нужно оставлять историю репликации. Значения описаны в разделе «[Метод `ClearReplicationProtocol`](#)».

5.34.28. Метод `GetDefaultConflictResolveRules`

Запрос правил разрешения конфликтов по умолчанию, настроенных на филиале. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. Подробнее о правилах разрешения конфликтов по умолчанию написано в разделе «Настройка правил автоматической обработки конфликтов» документа «Бастион-3 – Репликация. Руководство администратора».

Пример ответа на запрос:

```
{
  "default_conflict_resolve_rules": {
    "from_branch_to_center_rules": [
      {
        "entity_type_code": 1,
        "conflict_type": "CONFLICT_TYPE_VERSION",
        "rule": "CONFLICT_RESOLVE_TYPE_LEAVE_CONFLICT",
        "conflict_direction": "CONFLICT_DIRECTION_FROM_BRANCH_TO_CENTER",
        "entity_parameters": null
      },
      {
        "entity_type_code": 1,
        "conflict_type": "CONFLICT_TYPE_DUPLICATE",
        "rule": "CONFLICT_RESOLVE_TYPE_LEAVE_CONFLICT",
        "conflict_direction": "CONFLICT_DIRECTION_FROM_BRANCH_TO_CENTER",
        "entity_parameters": null
      },
      ...
    ],
    "from_center_to_branch_rules": [
      {
        "entity_type_code": 1,
        "conflict_type": "CONFLICT_TYPE_VERSION",
        "rule": "CONFLICT_RESOLVE_TYPE_LEAVE_CONFLICT",
        "conflict_direction": "CONFLICT_DIRECTION_FROM_CENTER_TO_BRANCH",
        "entity_parameters": null
      },
      {
        "entity_type_code": 1,
        "conflict_type": "CONFLICT_TYPE_DUPLICATE",
        "rule": "CONFLICT_RESOLVE_TYPE_LEAVE_CONFLICT",
        "conflict_direction": "CONFLICT_DIRECTION_FROM_CENTER_TO_BRANCH",
        "entity_parameters": null
      },
      ...
    ]
  }
}
```

Ответное сообщение содержит правила разрешения конфликтов по умолчанию, сгруппированные по направлениям изменений репликации – исходящие/входящие изменения. О каждом правиле представлена следующая информация:

- `entity_type_code` – код типа сущностей (см. раздел «[Коды типов сущностей репликации](#)»);
- `conflict_type` – тип конфликта:
 - `CONFLICT_TYPE_VERSION` – конфликт версий;

- CONFLICT_TYPE_DUPLICATE – конфликт, возникающий при нахождении дубликата сущности;
- rule – способ разрешения конфликта:
 - CONFLICT_RESOLVE_TYPE_LEAVE_CONFLICT – оставить конфликт данных;
 - CONFLICT_RESOLVE_TYPE_AVOID_CHANGE – отменить применение изменения;
 - CONFLICT_RESOLVE_TYPE_APPLY_CENTER_VERSION – применить версию с центра репликации;
 - CONFLICT_RESOLVE_TYPE_OVERRIDE_CENTER_VERSION – переопределить значение в центре репликации;
 - CONFLICT_RESOLVE_TYPE_MERGE_ENTITIES – объединить записи;
 - CONFLICT_RESOLVE_TYPE_CREATE_DUPLICATE – создать дубликат записи;
- conflict_direction – направление изменения:
 - CONFLICT_DIRECTION_FROM_BRANCH_TO_CENTER – изменение, отправленное с филиала на центр;
 - CONFLICT_DIRECTION_FROM_CENTER_TO_BRANCH – изменение, полученное филиалом с центра;
- entity_parameters – параметры для идентификации сущности, используются при ручном разрешении конфликтов. Для правил разрешения конфликтов по умолчанию всегда будет null.

5.34.29. Метод GetForeignDataEditSettings

Запрос параметров редактирования "чужих" данных, полученных филиалом через репликацию. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. Подробнее о параметрах редактирования "чужих" данных написано в разделе «Настройка параметров редактирования «чужих» данных» документа «Бастион-3 – Репликация. Руководство администратора».

Пример ответа на запрос:

```
{
  "settings": {
    "can_change_foreign_organization_structure": true,
    "can_change_foreign_passes": true,
    "replicate_foreign_data_changes": true
  }
}
```

Ответное сообщение содержит параметры редактирования "чужих" данных, настроенные на филиале:

- can_change_foreign_organization_structure – разрешать локально изменять "чужие" узлы организационной структуры, полученные через репликацию;
- can_change_foreign_passes – разрешать локально изменять данные "чужих" пропусков, полученных через репликацию;
- replicate_foreign_data_changes – реплицировать изменения "чужих" данных, полученных через репликацию.

5.34.30. Метод `GetGlobalAccessLevels`

Запрос списка глобальных уровней доступа. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "global_access_levels": [
    {
      "id": 1,
      "name": "Глобальный уровень доступа",
      "linked_local_access_level_id": {
        "value": 1
      }
    },
    {
      "id": 2,
      "name": "Глобальный уровень доступа 2",
      "linked_local_access_level_id": null
    },
    {
      "id": 3,
      "name": "Глобальный уровень доступа 3",
      "linked_local_access_level_id": null
    }
  ]
}
```

Ответное сообщение содержит список всех глобальных уровней доступа, имеющих на филиале. О каждом из них представлена следующая информация:

- `id` – идентификатор глобального уровня доступа на филиале;
- `name` – имя глобального уровня доступа;
- `linked_local_access_level_id` – идентификатор уровня доступа, который соответствует данному глобальному уровню доступа на филиале.

5.34.31. Метод `GetIsInReplication`

Выполняет проверку, находится ли филиал в репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа на запрос:

```
{
  "is_in": true
}
```

- `is_in` – признак, указывающий, что филиал введён в систему репликации.

5.34.32. Метод `GetPassCategoryOrganizationReplicationProperties`

Запрос настроек репликации по умолчанию для персональных пропусков указанной категории и организации/подразделения. В качестве параметров запроса указываются идентификатор категории пропусков и идентификатор организации/подразделения. Запрос может быть выполнен только с использованием токена доступа. Подробнее о настройках репликации по умолчанию написано в разделе «Настройка параметров репликации пропусков по умолчанию для подразделений и категорий пропусков» документа «Бастион-3 – Репликация. Руководство администратора».

Пример запроса:

```
{
  "pass_category_id": 2,
  "organization_node_id": 101
}
```

- `pass_category_id` – идентификатор категории пропусков;
- `organization_node_id` – идентификатор организации/подразделения.

Пример ответа на запрос:

```
{
  "replication_properties": {
    "target_branch_ids": [
      100,
      101
    ],
    "target_branch_custom_global_access_level_map": {
      "100": 2
    },
    "pass_category_id": 2,
    "organization_node_id": 101,
    "global_access_level_id": 1
  }
}
```

- `pass_category_id` – идентификатор категории пропусков;
- `organization_node_id` – идентификатор организации/подразделения;
- `global_access_level_id` – идентификатор глобального уровня доступа;
- `target_branch_ids` – идентификаторы филиалов – пунктов назначения;
- `target_branch_custom_global_access_level_map` – словарь отдельных назначений глобальных уровней доступа для филиалов – пунктов назначения. В филиале, идентификатор которого указан в качестве ключа, для пропусков будет действовать глобальный уровень доступа, идентификатор которого указан в качестве значения.

5.34.33. Метод `GetPassesOwnerUids`

Запрос глобальных идентификаторов филиалов – владельцев для списка персональных пропусков. В качестве параметра запроса указываются идентификаторы пропусков. Запрос может быть выполнен только с использованием токена доступа.

Пример запроса:

```
{
  "pass_ids": [101,102]
}
```

- pass_ids – список идентификаторов персональных пропусков, для которых запрашиваются филиалы – владельцы.

Пример ответа:

```
{
  "pass_owner_uid_map": {
    "101": "619fb5c0-3082-4ada-b1a3-a7c4483d2b5a",
    "102": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
  }
}
```

Ответное сообщение содержит в себе словарь, где ключ – идентификатор персонального пропуска, значение – глобальный идентификатор филиала – владельца пропуска.

5.34.34. Метод `GetPassReplicationProperties`

Запрос настроек репликации персонального пропуска. В качестве параметра запроса указывается идентификатор пропуска. Запрос может быть выполнен только с использованием токена доступа. Подробнее о настройках репликации пропусков написано в разделе «Редактирование настроек репликации пропуска» документа «Бастион-3 – Репликация. Руководство администратора».

Ответ на запрос включает в себя:

1. глобальный уровень доступа, назначенный пропуску;
2. филиалы – пункты назначения пропуска;
3. отдельные назначения глобальных уровней доступа для филиалов – пунктов назначения.

Пример запроса:

```
{
  "pass_id": 101
}
```

- pass_id – идентификатор персонального пропуска.

Пример ответа на запрос:

```
{
  "replication_properties": {
    "target_branch_ids": [
      100,
      101,
    ]
  }
}
```

```

        102,
        103
    ],
    "target_branch_custom_global_access_level_map": {
        "101": 2,
        "103": 3
    },
    "pass_id": 101,
    "global_access_level_id": {
        "value": 1
    }
}
}

```

- pass_id – идентификатор персонального пропуска;
- global_access_level_id – идентификатор глобального уровня доступа, назначенного пропуску;
- target_branch_ids – идентификаторы филиалов – пунктов назначения пропуска;
- target_branch_custom_global_access_level_map – словарь отдельных назначений глобальных уровней доступа для филиалов – пунктов назначения. В филиале, идентификатор которого указан в качестве ключа, для пропуска будет действовать глобальный уровень доступа, идентификатор которого указан в качестве значения.

5.34.35. Метод GetReplicationMetrics

Запрос метрик филиала репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. Подробнее о мониторинге состояния репликации написано в разделе «Мониторинг состояния системы репликации» документа «Бастион-3 – Репликация. Руководство администратора».

Пример ответа на запрос:

```

{
  "metrics": {
    "next_replication_date": {
      "seconds": "1765249200",
      "nanos": 0
    },
    "next_clear_replication_protocol_date": {
      "seconds": "1767164700",
      "nanos": 0
    },
    "branch_new_changes_count": 4,
    "branch_conflict_changes_count": 1,
    "branch_error_changes_count": 2,
    "center_new_changes_count": 12,
    "center_conflict_changes_count": 2,
    "center_error_changes_count": 0
  }
}

```

Ответное сообщение содержит следующие метрики филиала:

- `next_replication_date` – планируемая дата и время следующего запуска репликации по расписанию;
- `next_clear_replication_protocol_date` – планируемая дата и время следующего запуска процесса очистки протокола репликации по расписанию;
- `branch_new_changes_count` – количество новых исходящих изменений;
- `branch_conflict_changes_count` – количество конфликтных исходящих изменений;
- `branch_error_changes_count` – количество изменений с ошибкой при передаче в центр;
- `center_new_changes_count` – количество новых входящих изменений;
- `center_conflict_changes_count` – количество конфликтных входящих изменений;
- `center_error_changes_count` – количество входящих изменений с ошибкой применения.

5.34.36. Метод `GetReplicationScheduleSettings`

Запрос параметров выполнения репликации по расписанию на филиале. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа. Подробнее о расписании репликации написано в разделе «Настройка расписаний» документа «Бастион-3 – Репликация. Руководство администратора».

Пример ответа на запрос:

```
{
  "settings": {
    "is_enabled": true,
    "frequency": "REPLICATION_SCHEDULE_FREQUENCY_ONE_DAY",
    "start_time": {
      "hours": 20,
      "minutes": 0,
      "seconds": 0,
      "nanos": 0
    }
  }
}
```

Ответ на запрос включает в себя следующие параметры:

- `is_enabled` – признак, указывающий, включено ли выполнение репликации по расписанию;
- `frequency` – частота выполнения репликации:
 - `REPLICATION_SCHEDULE_FREQUENCY_ONE_DAY` – раз в день;
 - `REPLICATION_SCHEDULE_FREQUENCY_ONE_HOUR` – раз в час;
 - `REPLICATION_SCHEDULE_FREQUENCY_FIFTEEN_MINUTES` – раз в 15 минут;
 - `REPLICATION_SCHEDULE_FREQUENCY_FIVE_MINUTES` – раз в 5 минут;
 - `REPLICATION_SCHEDULE_FREQUENCY_ONE_MINUTE` – ежеминутно;
- `start_time` – время первого запуска репликации по расписанию в UTC.

5.34.37. Метод `IsInReplicationChanged`

Запрос на отслеживание события о введении/выведении филиала из системы репликации. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа при наступлении события:

```
{
  "is_in": false
}
```

- `is_in` – текущее состояние филиала: введён в репликацию или нет.

5.34.38. Метод `NewBranchChangeCreated`

Запрос на получение уведомлений о создании нового исходящего изменения на филиале. Выполняется с пустым телом запроса. Ответ содержит сообщение с пустым телом. Запрос может быть выполнен только с использованием токена доступа.

5.34.39. Метод `NewCenterChangesReceived`

Запрос на получение уведомлений о получении филиалом новых изменений с центра репликации. Выполняется с пустым телом запроса. Ответ содержит сообщение с пустым телом. Запрос может быть выполнен только с использованием токена доступа.

5.34.40. Метод `NextClearReplicationProtocolDateChanged`

Запрос на получение уведомлений об обновлении планируемой даты и времени следующего запуска выполнения очистки протокола репликации по расписанию. Выполняется с пустым телом запроса. Ответ содержит сообщение с пустым телом. Запрос может быть выполнен только с использованием токена доступа.



Планируемую дату и время следующего запуска выполнения очистки протокола репликации по расписанию можно получить с помощью запроса метрик филиала репликации (см. раздел «[GetReplicationMetrics](#)»).

5.34.41. Метод `NextReplicationDateChanged`

Запрос на получение уведомлений об обновлении планируемой даты и времени следующего запуска репликации по расписанию. Выполняется с пустым телом запроса. Ответ содержит сообщение с пустым телом. Запрос может быть выполнен только с использованием токена доступа.



Планируемую дату и время следующего запуска репликации по расписанию можно получить с помощью запроса метрик филиала репликации (см. раздел «[GetReplicationMetrics](#)»).

5.34.42. Метод `OrganizationOwnerChanged`

Запрос на отслеживание события об изменении филиала – владельца организации/подразделения. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа при наступлении события:

```
{
  "organization_node_info": {
    "organization_node_id": 106,
    "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
  }
}
```

Ответное сообщение содержит следующие параметры события:

- `organization_node_id` – идентификатор организации/подразделения на филиале;
- `owner_uid` – глобальный идентификатор нового владельца организации/подразделения – участника репликации.

5.34.43. Метод `PassOwnerChanged`

Запрос на отслеживание события об изменении филиала – владельца персонального пропуска. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа при наступлении события:

```
{
  "pass_info": {
    "pass_id": 106,
    "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"
  }
}
```

Ответное сообщение содержит следующие параметры события:

- `pass_id` – идентификатор персонального пропуска на филиале;
- `owner_uid` – глобальный идентификатор нового владельца пропуска – участника репликации.

5.34.44. Метод `PersonOwnerChanged`

Запрос на отслеживание события об изменении филиала – владельца персоны. Выполняется с пустым телом запроса. Запрос может быть выполнен только с использованием токена доступа.

Пример ответа при наступлении события:

```
{
  "person_info": {
    "person_id": 107,
  }
}
```

```
    "owner_uid": "a80e5512-5eee-4c93-844a-d315c2f94c4a"  
  }  
}
```

Ответное сообщение содержит следующие параметры события:

- person_id – идентификатор персоны на филиале;
- owner_uid – глобальный идентификатор нового владельца персоны – участника репликации.

5.34.45. Метод RefreshCenterChanges

Запрос на загрузку новых изменений с центра репликации. Загруженные с центра изменения сохраняются на филиале. Запрос выполняется с пустым телом запроса. Ответ на запрос – сообщение с пустым телом. Запрос может быть выполнен только с использованием токена доступа.

5.34.46. Метод ReplicationProtocolCleaned

Запрос на получение уведомлений об очистке протокола репликации на филиале. Выполняется с пустым телом запроса. Уведомление поступает в случае успешного выполнения очистки, при этом ответ содержит сообщение с пустым телом. Запрос может быть выполнен только с использованием токена доступа.

6. Приложения

6.1. Приложение 1. Тип google.protobuf.Any

Тип Any может служить для описания любого типа сообщения в protobuf, что позволяет расширять возможности протокола, не изменяя описания базовых сообщений. Поле сообщения с типом Any представляет из себя по сути вложенную protobuf-сериализацию другого сообщения. Для передачи через сообщение Any любого другого сообщения его необходимо сначала "запаковать". Для получения сообщения из Any его нужно "распаковать" явно указав тип сообщения, которое должно быть десериализовано.

Поля типа Any:

- `type_url` – строка, идентифицирующая тип запакованного сообщения. Значение в общем случае формируется по следующему шаблону: `type.googleapis.com/[имя пакета].[Тип запакованного сообщения]`, где имя пакета соответствует значению указанному в поле `package` proto-файла, в котором приведено описание типа запакованного сообщения.
- `value` – base64-строка, в которой содержится массив байт соответствующий protobuf-сериализации запакованного сообщения.

6.2. Приложение 2. Ошибки, возвращаемые ПК «Бастион-3»

Полный перечень возможных кодов ошибок gRPC, возвращаемых ПК «Бастион-3», и их описание приведены ниже. Ошибки со стороны сервера системы, не приведенные в данной таблице, по умолчанию будут возвращены с кодом 13 INTERNAL.

Таблица 2. Коды ошибок ПК "Бастион-3"

Код ошибки gRPC	Код ошибки на сервере системы	Описание
3 INVALID_ARGUMENT	System.ArgumentException	Некорректное значение параметра метода.
	-17 IncorrectDataError	Некорректные данные или состояние объекта.
	-19 UnsupportedError	Неизвестный фильтр поиска объектов или некорректные параметры фильтра.
5 NOT_FOUND	-10 NotFoundError	Запрошенные данные не были найдены.
7 PERMISSION_DENIED	-14 Forbidden	Недостаточно прав для запрошенной информации.
13 INTERNAL	-1 ConnectionError	Ошибка подключения.

	-2 ProtocolError	Ошибка протокола передачи данных.
	-3 WrongFormatError	Неправильный формат представления данных.
	-5 LogicError	Ошибка бизнес-логики.
	-6 DbError	Ошибка базы данных.
	-7 RejectedError	Запрос или команда была отброшена.
	-13 UndefinedError	Неопределённая ошибка.
	-16 LicenseError	Ошибка лицензирования.
	-19 UnsupportedError	Запрос выполнения действия, которое не поддерживается.
	-20 TimeoutError	Тайм-аут операции.
	-21 InitializationError	Ошибка инициализации приложения.
	-22 ExternalSystemError	Ошибка внешней системы.
16 UNAUTHENTICATED	-9 AuthorizationError	Ошибка авторизации.
	-12 UnauthorizedAccessError	Доступ к защищённым данным или попытка выполнения действия без авторизации.

6.3. Приложение 3. Основные элементы фильтрации

Приложение содержит описание базовых элементов фильтрации, используемых в механизмах поиска пропусков и устройств через gRPC. Данные элементы служат стандартным способом задания условий поиска для различных типов полей.

- **Int32SearchTermEntry** – элемент фильтрации, позволяющий задать критерии для целочисленного параметра. Поддерживает операции сравнения и поиск в диапазоне.

Поля:

- **from** (`google.protobuf.Int32Value`) – нижняя граница. Если задана, то будут возвращены все сущности, значение параметра которых больше или равно значению нижней границы.;
- **to** (`google.protobuf.Int32Value`) – верхняя граница. Если задана, то будут возвращены все сущности, значение параметра которых меньше или равно значению нижней границы.

- **AssignedSearchTermEntry** – элемент фильтрации, позволяющий задать критерии поиска по тому, имеет или нет значение конкретный параметр.

Поля:

- **is_assigned** – булево значение, указывающее наличие заданного параметра:
 - если параметр имеет значение *true*, то будут возвращены элементы, у которых параметр задан;
 - если параметр имеет значение *false*, то будут возвращены элементы, у которых параметр не задан.
- **TimestampSearchTermEntry** – элемент фильтрации, позволяющего задать критерии для временного параметра, используется для описания диапазона времени (обязательно должна быть указана хотя бы одна из границ).

Поля:

- **from** (`google.protobuf.Timestamp`) – нижняя граница времени. Если задана, то будут возвращены все сущности, значение параметра которых больше или равно значению нижней границы;
- **to** (`google.protobuf.Timestamp`) – верхняя граница времени. Если задана, то будут возвращены все сущности, значение параметра которых меньше значения верхней границы.
- **TimestampValueSearchTermEntry** – позволяет задать условие для необязательного временного параметра. Содержит один из двух вариантов:
 - **assigned** – проверка наличия значения (с использованием `AssignedSearchTermEntry`);
 - **range** – фильтрация по диапазону (с использованием `TimestampSearchTermEntry`).
- **DateSearchTermEntry** – элемент фильтрации, позволяющего задать критерии для параметра-даты.

Поля:

- **from** – нижняя граница диапазона. Если задана, то будут возвращены все сущности, значение параметра которых больше или равно значению нижней границы.;
- **to** – верхняя граница диапазона. Если задана, то будут возвращены все сущности, значение параметра которых меньше значения верхней границы.
- **DateValueSearchTermEntry** – элемент фильтрации аналогичен `TimestampValueSearchTermEntry`, но для даты. Позволяет либо проверить, задана ли дата, либо отфильтровать по диапазону дат.
- **TimeOfDaySearchTermEntry** – элемент фильтрации используется для описания диапазона времени дня (обязательно должна быть указана хотя бы одна из границ диапазона. Если не указана нижняя, то берутся записи произошедшие с начала дня до верхней границы, и, если не указана верхняя граница, то берутся все записи начиная с нижней границы до конца дня).

Поля:

- from – нижняя граница времени дня (включается в результаты запроса);
- to – верхняя граница времени дня (не включается в результаты запроса).
- **IdsSearchTermEntry** – элемент фильтрации позволяет задать список целочисленных идентификаторов, по которым выполняется поиск.

Поля:

- ids (repeated int32) – массив значений.
- **IdsValueSearchTermEntry** – элемент предоставляет возможность фильтрации по необязательному параметру-идентификатору. Содержит один из двух вариантов:
 - assigned – проверка наличия значения (с использованием AssignedSearchTermEntry);
 - ids – проверка на вхождение в заданный список (с использованием IdsSearchTermEntry).
- **StringIdsSearchTermEntry** – элемент предназначен для фильтрации по коллекции строковых идентификаторов.

Поля:

- ids (repeated string) – список строковых идентификаторов.
- **StringSearchTermEntry** – элемент фильтрации позволяет определять условие поиска по строковому параметру с использованием подстроки.

Поля:

- substring (google.protobuf.StringValue) – подстрока, по которой производится поиск;
- filter_operator – тип операции фильтрации. Возможные операции фильтрации:
 - STRING_FILTER_OPERATOR_UNSPECIFIED (0) – не указан (интерпретируется как CONTAINS);
 - STRING_FILTER_OPERATOR_CONTAINS (1) – содержит строку;
 - STRING_FILTER_OPERATOR_STARTS_WITH (2) – начинается со строки;
 - STRING_FILTER_OPERATOR_ENDS_WITH (3) – заканчивается строкой;
 - STRING_FILTER_OPERATOR_EQUAL (4) – равняется строке.
- case_sensitive (bool) – флаг учета регистра при поиске (опциональный параметр).
- **StringValueSearchTermEntry** – элемент используется для задания фильтра по необязательному строковому параметру. Реализует один из следующих вариантов:
 - assigned – проверка наличия строки (с использованием AssignedSearchTermEntry);
 - substring – строка для простого поиска по вхождению подстроки без учёта регистра;
 - searchString – сложное условие поиска (с использованием StringSearchTermEntry).

- **Int64SearchTermEntry** – элемент фильтра для задания диапазона для 64-битных целых чисел:

Поля:

- **from** (google.protobuf.Int64Value) – нижняя граница. Если задана, то будут возвращены все сущности, значение параметра которых больше или равно значению нижней границы;
- **to** (google.protobuf.Int64Value) – верхняя граница. Если задана, то будут возвращены все сущности, значение параметра которых меньше значения верхней границы.

- **Int64ValueSearchTermEntry** – элемент фильтрации аналогичен Int64SearchTermEntry, но для необязательного параметра. Может принимать один из двух вариантов:

- **assigned** – проверка наличия значения (с использованием AssignedSearchTermEntry);
- **range** – поиск по диапазону (с использованием Int64SearchTermEntry).

- **DoubleSearchTermEntry** – элемент фильтрации предназначен для задания диапазона для значений с плавающей запятой.

Поля:

- **from** (google.protobuf.DoubleValue) – нижняя граница. Если задана, то будут возвращены все сущности, значение параметра которых больше или равно значению нижней границы;
- **to** (google.protobuf.DoubleValue) – верхняя граница. Если задана, то будут возвращены все сущности, значение параметра которых меньше значения верхней границы.

- **DoubleValueSearchTermEntry** – элемент фильтрации по необязательному параметру с плавающей запятой. Позволяет проверить, задано ли значение, или выполнить фильтрацию по диапазону (с использованием DoubleSearchTermEntry).

- **BooleanValueSearchTermEntry** – элемент для фильтрации по логическому параметру. Реализует один из вариантов:

- **assigned** – проверка наличия логического значения (с использованием AssignedSearchTermEntry);
- **value** – конкретное логическое значение для фильтрации.

6.4. Приложение 4. Состояния устройств

Полный перечень возможных кодов состояний устройств и их расшифровка приведены ниже. Большая часть этих состояний применяется только к ограниченному числу типов устройств (например, все состояния «с ограничением доступа» – применяются только для точек прохода).

Код	Состояние	Тип состояния
0	Неизвестно	Штатное

1	Норма	Штатное
2	Недоступно	Штатное
3	Не активно	Штатное
4	Тревога	Тревога
5	Неисправность	Неисправность
6	Тревога при входе	Тревога
7	Тревога при выходе	Тревога
8	Тревога при входе с ограничением доступа	Тревога
9	Тревога при выходе с ограничением доступа	Тревога
10	Взлом	Тревога
11	Взлом при ограничении доступа	Тревога
12	Штатный вход	Штатное
13	Выполняется вход при ограничении доступа	Штатное
14	Заблокировано	Штатное
15	Не активно при ограничении доступа	Штатное
16	Разблокировано при ограничении доступа	Штатное
17	Разблокировано	Штатное
18	Разблокировано, закрыто	Штатное
19	Разблокировано при ограничении доступа	Штатное
20	Штатный выход	Штатное
21	Выполняется выход при ограничении доступа	Штатное

22	Неисправность при закрытии	Неисправность
23	Неисправность при закрытии в режиме ограничения доступа	Неисправность
24	Удержание	Неисправность
25	Удержание (доступ ограничен)	Неисправность
26	Тревога в полуоткрытом состоянии	Тревога
27	Тревога в полуоткрытом состоянии (доступ ограничен)	Тревога
28	Тревога в незапертом состоянии	Тревога
29	Приоткрыто	Штатное
30	Заблокировано, дверь открыта	Штатное
31	Включено	Штатное
32	Выключено	Штатное
33	Неготовность	Неисправность
34	Активно	Штатное
35	Точка прохода заблокирована	Штатное
36	Дверь не заперта	Тревога
37	Выполняется вход под принуждением	Тревога
38	Выполняется выход под принуждением	Тревога
39	Тревога и неисправность одновременно	Тревога
40	Штатный проход (дверь открыта)	Штатное
41	Видеозапись включена	Штатное
42	Заблокировано на вход	Штатное

43	Заблокировано на выход	Штатное
44	Разблокировано на вход	Штатное
45	Разблокировано на выход	Штатное
46	Разблокировано на вход, заблокировано на выход	Штатное
47	Заблокировано на вход, разблокировано на выход	Штатное
48	Частично на охране	Штатное
49	Предтревога (предупреждение, используется для периметров)	Тревога
50	Отключено	Штатное