Контроль активности пропуска

Часто бывает полезно выявить неактивные пропуска, то есть пропуска, по которым давно не было никаких событий доступа. Для решения этой задачи в Платформе НЕЙРОСС реализован скрипт автоматизации, позволяет автоматически обнаружить неактивные пропуска и поместить их в отдельную папку, а также, при необходимости, — выполнить бессрочную приостановку пропуска.

Контролируются не все пропуска, а только пропуска выбранных владельцев.

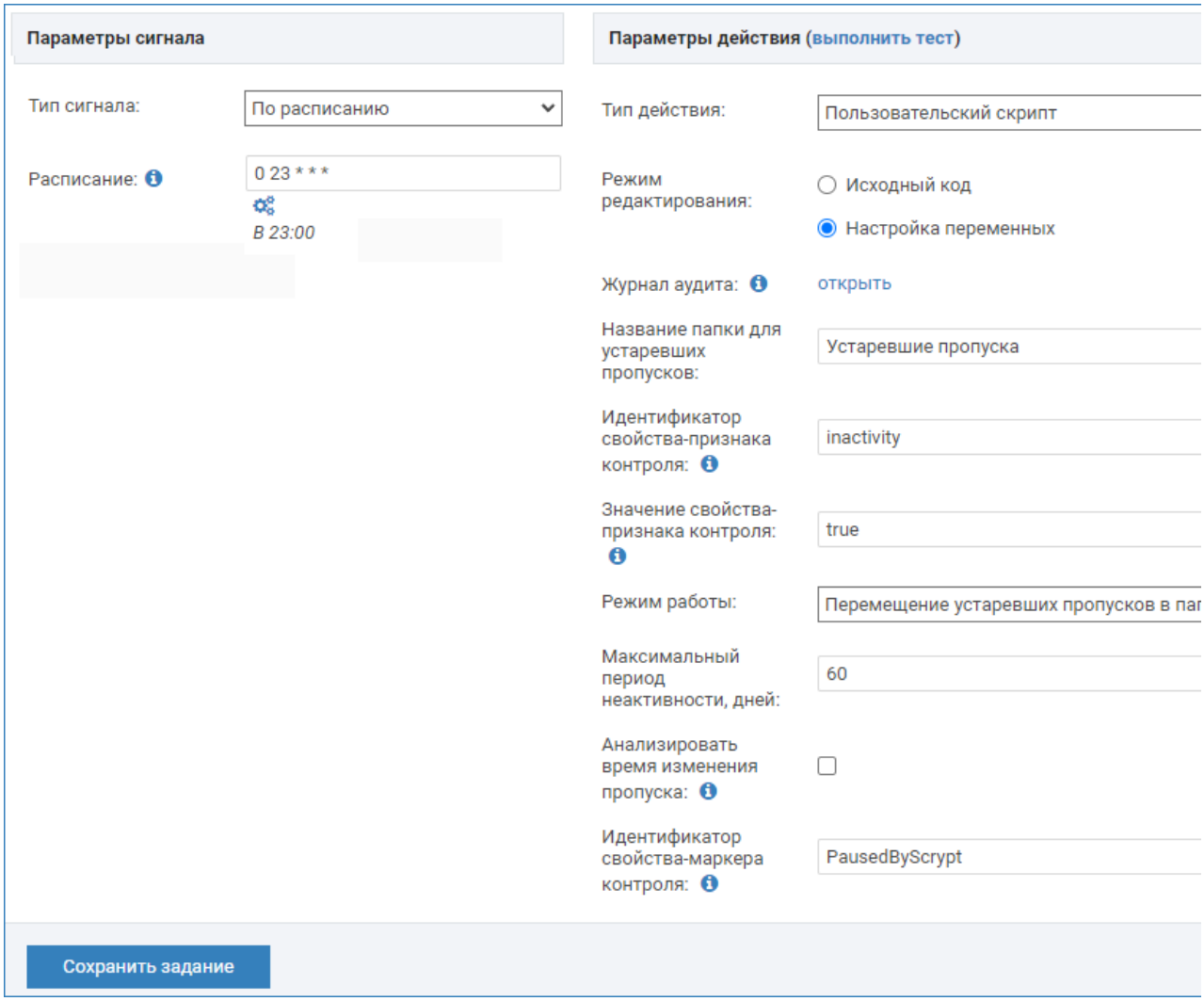
Посредством настройки свойств скрипта автоматизации вы можете:

- задать имя папки для помещения пропусков;
- определить, требуется ли автоматически выполнять изъятие пропусков, либо решение будет принимать оператор бюро пропусков;
- задать период неактивности — временной интервал в днях, по истечении которого при отсутствии событий доступа пропуск считается неактивным;
- указать, требуется ли учитывать факт изменения пропуска совместно с событием доступа — пропуск будет считаться неактивным, если за заданный период не было ни события доступа, ни факта изменения пропуска оператором бюро пропусков;
- пометить, что пропуск приостановлен скриптом автоматизации.

> ⓘ Задание автоматизации может запускаться оператором вручную, по расписанию (например, раз в неделю по воскресеньям), либо по какому-либо событию/системному действию.

Пример: Найти пропуска, по которым не было событий доступа в течение 60 дней и поместить их в папку «Устаревшие пропуска»

Задание автоматизации запускается автоматически ежедневно в 23:00, производится поиск за интервал в 60 дней, пропуска не приостанавливаются, но помещаются в папку «Устаревшие пропуска», факт изменения пропуска не анализируется.



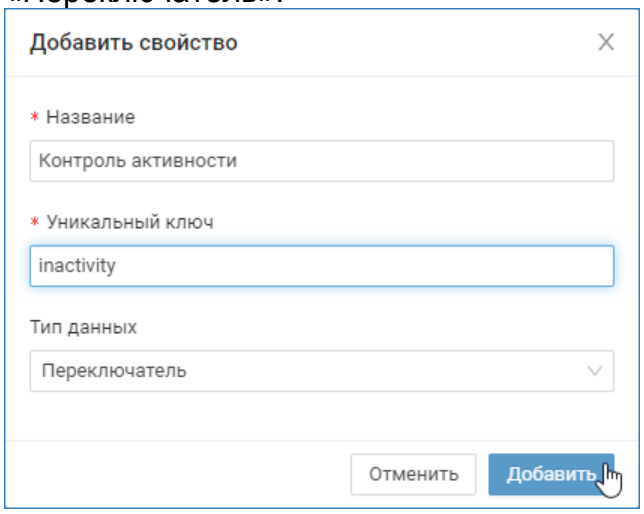
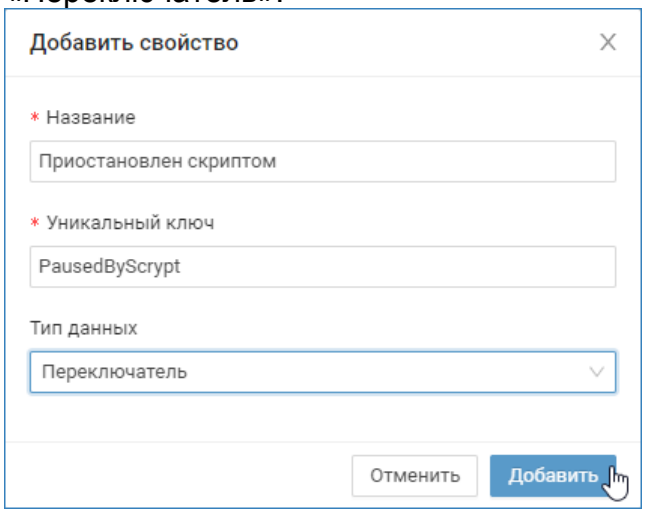
Настройки СКУД

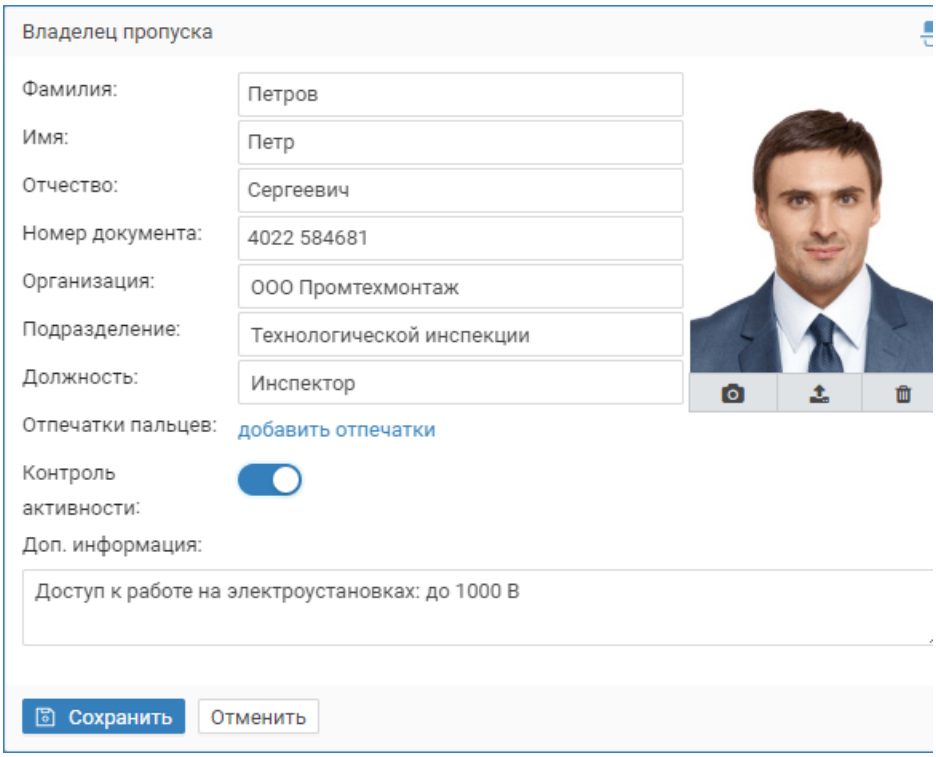
Вам потребуется:

1. Создать пользовательское свойство владельца пропуска для указания перечня лиц, для которых требуется проводить контроль активности пропуска.
2. Добавить свойство на форму владельца пропуска и задать в значение true для владельцев, чьи пропуска требуется анализировать.
3. Создать пользовательское свойство пропуска для пометок о том, что пропуск приостановлен скриптом.
4. Добавить свойство на форму пропуска.

Порядок действий:

Задача	Комментарий

Создание пользовательского свойства	В разделе **Настройки СКУД** АРМ НЕЙРОСС Доступ: 1. На вкладке **Свойства владельца пропуска** добавьте новое свойство, задайте название и ключ свойства, укажите тип «Переключатель».  2. На вкладке **Свойства пропуска** добавьте новое свойство, задайте название и ключ свойства, укажите тип «Переключатель».  Добавление пользовательских свойств

Изменение формы владельца пропуска	В разделе **Персонал** АРМ НЕЙРОСС Доступ выберите папку пропусков, для которой нужно настроить контроль активности перейдите к вкладке **Настройки папки**: 1. На вкладке **Форма ввода: Владелец пропуска** добавьте пол **Контроль активности** (название может быть любым) на фор владельца пропуска. 2. На вкладке **Форма ввода: Пропуск добавьте** поле **Приостановлен скриптом** (название может быть любым) на форму пропуска. Рекомендуем использовать Конструктор форм. При использовании нестандартных форм, обратитесь к специалистам компании ИТРИУМ за услугой по доработк формы. Настройка форм ввода данных При необходимости вывода информации в таблице пропусков вкладке **Настройка таблицы** добавьте вывод данных свойств. Настройка таблицы пропусков
Включение необходимости контроля активности пропуска	Выберите владельцев, для которых требуется проводить контроль активности пропуска, и в режиме редактирования пропуска / группы пропусков в поле **Контроль активности** установите переключатель в положение **Включено**. Сохраните данные владельца.  Управление пропусками

Просмотр результатов работы скрипта	Дождитесь выполнения условия запуска задания (в нашем примере это 23:00 текущего дня).

Настройки автоматизации

Скопируйте приведённый ниже код скрипта и создайте файл с произвольным именем и расширением **JSON**, например: *задание_автоматизации_Контроль_устаревших пропусков.json*. Для этого удобно использовать простые текстовые редакторы типа Блокнот или Notepad++. Вы также можете обратиться к специалистам компании ИТРИУМ, мы вышлем подготовленный файл.

```
import play.api.libs.json.Format
import beans.pacs.{PacsFolderBean, PassBean}
import extensions.automation.scripts.AutomationActionScript
import extensions.automation.signals.AutomationSignal
import models.common.UserIdentity
import models.neyross.{Pass, PassComponent}
import models.pacs.{PacsFolder, PacsFolderComponent}
import dto.pacs.PacsFolderMovementDto
import services.logging.web.{LoggerWithWeb => Logger}
import play.api.libs.json.{JsObject, Json}
import proto.neyross.PassProto
import services.common.{SystemLogService, SystemLogTag}
import services.neyross.NeyrossEmbedApiService
import slick.dbio.DBIOAction
import utils.TableExtension
import utils.bootstrap.CustomPostgresProfile
import utils.common.ScalaUtils

import java.time.OffsetDateTime
import java.util.UUID
import scala.concurrent.Future

class ObsoletePassControlAutomationScript extends AutomationActionScript {

  val logger = Logger("ObsoletePassControlAutomationScript")

  private val postgresProfile = ctx.dbConfig.profile.asInstanceOf
[CustomPostgresProfile]

  import postgresProfile.api._

  implicit val ec = ctx.executionContext
  private val neyrossEmbedApi = ctx.injector.instanceOf
[NeyrossEmbedApiService]
  private val pacsFolderBean = ctx.injector.instanceOf[PacsFolderBean]
```

```
    private val systemLogService = ctx.injector.instanceOf[SystemLogService]
    private val passBean = ctx.injector.instanceOf[PassBean]

    private val folders = new PacsFolderComponent()(ctx.dbConfig)
    private val passes = new PassComponent()(ctx.dbConfig, neyrossEmbedApi, ec)

    // @parameter { "type": "string", "title": "Название папки для устаревших
пропусков", "key": "workingFolder" }
    val workingFolder = "Устаревшие пропуска"

    // @parameter { "type": "string", "title": "Идентификатор свойства-признака
контроля", "key": "shouldControlOutdateProperty", "description":
"Контролироваться будут пропуска только тех владельцев, которые имеют
данное свойство" }
    val shouldControlOutdateProperty = "inactivity"

    // @parameter { "type": "string", "title": "Значение свойства-признака
контроля", "key": "shouldControlOutdatePropertyValue", "description":
"Контролироваться будут пропуска только тех владельцев, которые имеют
указанное значение данного свойства (true означает Да)" }
    val shouldControlOutdatePropertyValue = "true"

    // @parameter { "type": "select", "title": "Режим работы", "key": "mode",
"options": [ { "value": "default", "label": "Перемещение устаревших пропусков в
папку" }, { "value": "withStopping", "label": "Приостановка и перемещение
устаревших пропусков в папку" } ] }
    val mode = "withStopping"

    // @parameter { "type": "number", "title": "Максимальный период
неактивности, дней", "key": "inactivityPeriod" }
    val inactivityPeriod = 1

    // @parameter { "type": "boolean", "title": "Анализировать время изменения
пропуска", "key": "isUpdateTimeAnalysisEnabled", "description": "Если флаг
установлен, при расчёте активности пропуска будут учитываться также
события изменения пропуска оператором бюро пропусков. В противном случае
будет анализироваться только время последнего прохода по данному
пропуску" }
    val isUpdateTimeAnalysisEnabled = false

    // @parameter { "type": "string", "title": "Идентификатор свойства-маркера
контроля", "key": "scriptMarkPropertyKey", "description": "При автоматической
приостановке пропуска к нему будет добавлено свойство с данным
идентификатором и значением true (Да). Вы можете вынести данное свойство
на форму пропуска или в таблицу пропусков, при необходимости" }
    val scriptMarkPropertyKey = "PausedByScrypt"
```

```
    val modeLocalization: Map[String, String] =
      Map(
        "default" -> "Перемещение устаревших пропусков в папку",
        "withStopping" -> "Приостановка и перемещение устаревших пропусков в
папку"
      )

    logger.debug(s"mode: ${modeLocalization.getOrElse(mode, "")}")
    logger.debug(s"modification time analysing: $isUpdateTimeAnalysisEnabled")
    logger.debug(s"inactivity period: $inactivityPeriod")

    override def onSignal(signal: AutomationSignal): Future[Unit] = {
      workingFolder.trim match {
        case "" =>
          logger.warn("working folder name is empty. do nothing")
          Future.unit
        case nonEmptyFolderName =>
          runWithNonEmptyFolderName(nonEmptyFolderName)
      }
    }

    private def createFolderWithNameAndParent(name: String, parent: Option
[String]) = {
      val uuid = UUID.randomUUID().toString
      val folder = PacsFolder(
        id = None,
        uuid = Some(uuid),
        name = Some(name),
        parent = parent,
        isDefault = false,
        templates = None,
        defaultValues = None,
        settings = None,
        created = OffsetDateTime.now(),
        deleted = false
      )
      folders.insert(folder).map(_ => Some(uuid))
    }

    private def getFolderUuidByName(folderName: String) = {
      folders.folders
        .filter(f => f.name === folderName && !f.deleted)
        .result
        .headOption
        .flatMap({
          case Some(folder) =>
            DBIOAction.successful(folder.uuid)
```

```
          case _ =>
            folders
              .findDefault()
              .flatMap({
                case Some(defaultFolder) if defaultFolder.uuid.nonEmpty =>
                  createFolderWithNameAndParent(folderName, defaultFolder.uuid)
                case _ =>
                  logger.warn("there is no valid default folder. wtf?")
                  DBIOAction.successful(None)
              })
        })
    }

    private def isOutdated(timestamp: OffsetDateTime): Boolean = {
      val threshold = OffsetDateTime.now().minusDays(inactivityPeriod)
      logger.trace(s"checking if chosen timestamp $timestamp is older than
threshold $threshold")
      threshold.isAfter(timestamp)
    }

    private def isStoppedOrInOutdatedFolder(
                                            pass: Pass,
                                            outdatedFolderUuid: String,
                                            passFolders: List[String]
                                          ): Boolean = {
      pass.getState.contains(PassProto.Pass.State.sSTOPPED) || passFolders.
contains(outdatedFolderUuid)
    }

    private def isOutdatedPassRecord(record: PassRecord): Boolean = {
      // from all pass times, choose latest one and check if it outdated
      val lastAccessGrantedAtOption = record._2
      val created = record._3
      val lastUpdatedAtOption = record._4
      logger.trace(s"last access granted: $lastAccessGrantedAtOption, created:
$created, last updated: $lastUpdatedAtOption")
      // in any cases, pass has "created" time
      var latestTimeToCheck: OffsetDateTime = created
      if (
        lastAccessGrantedAtOption.isDefined &&
          lastAccessGrantedAtOption.get.isAfter(latestTimeToCheck)
      ) {
        // last access event was later than created time - using it
        latestTimeToCheck = lastAccessGrantedAtOption.get
      }
      if (isUpdateTimeAnalysisEnabled) {
        if (
```

```
          lastUpdatedAtOption.isDefined &&
            lastUpdatedAtOption.get.isAfter(latestTimeToCheck)
        ) {
          // last update event was later than created time and last access time - using it
          latestTimeToCheck = lastUpdatedAtOption.get
        }
      }
      isOutdated(latestTimeToCheck)
    }

    private def getPassesToMove(records: Seq[PassRecord], outdatedFolderUuid:
String): Seq[Pass] = {
      records
        .filter(record => {
          logger.trace(s"pass ${record._5.uuid} handling")
          !isStoppedOrInOutdatedFolder(record._5, outdatedFolderUuid, record._1.
getOrElse(Nil)) &&
            isOutdatedPassRecord(record)
        })
        .map(_._5)
    }

    type PassFolders = Option[List[String]]
    type LastAccessGrantedAt = Option[OffsetDateTime]
    type Created = OffsetDateTime
    type LastUpdatedAt = Option[OffsetDateTime]
    type PassRecord = (PassFolders, LastAccessGrantedAt, Created,
LastUpdatedAt, Pass)

    type PersonUUID = String
    type PersonFolders = Option[Seq[String]]
    type PersonRecord = (PersonUUID, PersonFolders)

    private def personToControlHandler(
                                  outdatedFolderUuid: String,
                                  personRecord: PersonRecord
                                ): Future[Seq[String]] = {
      val dbAction =
        passes
          .extendedPasses4(
            TableExtension.folders(ctx.dbConfig),
            TableExtension.lastAccessGrantedAt(ctx.dbConfig),
            TableExtension.created(ctx.dbConfig),
            TableExtension.lastUpdatedAt(ctx.dbConfig)
          )
          .filter(r => r.person === personRecord._1 && !r.deleted)
          .result
```

```
      ctx
        .dbRun(dbAction)
        .flatMap(result => {
          logger.trace(s"found passes of person ${personRecord._1}: ${result.map(_.
_5.uuid)}")
          val passesToMove = getPassesToMove(result, outdatedFolderUuid)
          logger.trace(s"found outdated passes of person ${personRecord._1}:
${passesToMove.map(_.uuid)}")
          if (passesToMove.nonEmpty) {
            val isPersonInOutdatedFolder = personRecord._2.exists(_.contains
(outdatedFolderUuid))
            val personsToMove = Option.when(!isPersonInOutdatedFolder)
(personRecord._1).toList
            val dto = PacsFolderMovementDto(personsToMove, passesToMove.map(_.
uuid))
            pacsFolderBean.copyToFolder(outdatedFolderUuid, dto)(ui = null, None).
map(_ => passesToMove.map(_.uuid))
          } else {
            Future.successful(Nil)
          }
        })
    }

    private def getPersonsToControl: Future[Vector[PersonRecord]] = {
      val personsToControl =
        sql"""
   SELECT DISTINCT uuid, folders
    FROM
        neyross_person,
        jsonb_to_recordset((person_json->>'properties')::jsonb) AS props(key text,
value text)
    WHERE
        key = '#${shouldControlOutdateProperty}' AND
        value = '#${shouldControlOutdatePropertyValue}' AND
        neyross_person.deleted = false
    """.as[PersonRecord]

      ctx.dbRun(personsToControl)
    }

    case class Property(key: String, value: String, index: Option[Long] = None)

    object Property {
      implicit val jsonFormat: Format[Property] = Json.format[Property]
    }
```

```
    private def stopPassesWithUuid(uuids: Seq[String]): Future[Unit] = {
      ctx
        .dbRun(passes.findByUuids(uuids))
        .flatMap(updatedPasses => {
          ScalaUtils
            .sequentialTraverse(updatedPasses)(updatedPass => {
              logger.trace(s"stopping pass ${updatedPass.uuid}; setting property
$scriptMarkPropertyKey and disabled_from fields")
              val passJson = updatedPass.toFullJson.asOpt[JsObject].get
              val passProperties = (passJson \ "properties").asOpt[List[Property]].
getOrElse(Nil)

              val newPassProperties = passProperties.filter(p => {
                p.key != scriptMarkPropertyKey
              }) :+ Property(
                scriptMarkPropertyKey,
                "true"
              )
              val newPassJson = passJson ++ Json.obj(
                "disabled_from" -> OffsetDateTime.now().toInstant.getEpochSecond.
toInt,
                "properties" -> Json.toJson(newPassProperties)
              )
              logger.trace(s"new pass json is: $newPassJson")
              passBean
                .updateV2(updatedPass.uuid, newPassJson)(UserIdentity.
getFakeUserIdentityFor("Автоматизация"), None)
            })
            .map(_ => ())
        })
    }

    private def runWithNonEmptyFolderName(folderName: String): Future[Unit] = {
      systemLogService
        .log(
          SystemLogTag.NORMAL,
          SystemLogTag.AUTOMATION
        )(s"Запущен процесс контроля устаревших пропусков")
        .map(_ => ())
      ctx
        .dbRun(getFolderUuidByName(folderName))
        .flatMap({
          case Some(folderUUID) =>
            logger.debug(s"working folder is $folderName ($folderUUID)")
            getPersonsToControl
              .flatMap(persons => {
                logger.debug(s"found persons to control: $persons")
```

```
                ScalaUtils.sequentialTraverse(persons)(personToControlHandler
(folderUUID, _))
              })
              .flatMap(uuids => {
                val flatUuidsList = uuids.flatten.distinct
                if (flatUuidsList.nonEmpty) {
                  val stopping = if (mode == "withStopping") {
                    stopPassesWithUuid(flatUuidsList)
                  } else {
                    Future.unit
                  }
                  stopping.flatMap(_ => {
                    val yesOrNo = Option
                      .when(isUpdateTimeAnalysisEnabled)("да")
                      .getOrElse("нет")
                    logger.debug(s"were moved ${flatUuidsList.size} passes")
                    val message =
                      s"""В папку '$folderName' было перемещено ${flatUuidsList.size}
пропусков.
                        |Режим работы: ${modeLocalization.getOrElse(mode, "")}.
                        |Анализировать время изменения пропуска: $yesOrNo
                        |Максимальный период неактивности: $inactivityPeriod дней""".
stripMargin
                    systemLogService
                      .log(
                        SystemLogTag.NORMAL,
                        SystemLogTag.AUTOMATION
                      )(
                        message = message,
                        data = Some(Json.toJson(flatUuidsList))
                      )
                      .map(_ => ())
                  })
                } else {
                  logger.debug("not found persons with obsolete persons or passes")
                  systemLogService
                    .log(
                      SystemLogTag.NORMAL,
                      SystemLogTag.AUTOMATION
                    )(s"Не найдено владельцев для контроля устаревших пропусков,
либо все пропуска таких владельцев не являются устаревшими")
                    .map(_ => ())
                }
              })
          case _ =>
            Future.unit
        })
```

```
    Future.unit
  }
}

new ObsoletePassControlAutomationScript
```

Добавьте задание автоматизации

1. В разделе **Автоматизация** нажмите на кнопку **Добавить новое задание**, нажмите на кнопку **Импорт** и укажите путь к подготовленному на предыдущем этапе файлу [Автоматизация].
2. В блоке **Параметры сигнала** выберите **По расписанию**, настройте расписание (например, 0 23 * * * для ежедневного запуска, или 0 23 * * 1 для запуска раз в неделю по воскресеньям).
3. В блоке **Параметры действия** настройте параметры задания автоматизации согласно таблице ниже.

Параметр	Комментарий
Название папки устаревших пропусков	Название папки. При запуске скрипта проверяется наличие папки с указанным именем в папке **Все**. При отсутствии, — она будет создана. В эту папку будут помещаться все найденные пропуска и их владельцы. Если название пустое, задание не выполняется.
Идентификатор свойства — признака контроля	Уникальный идентификатор пользовательского свойства владельца пропуска, созданный на этапе Настройки СКУД. В нашем примере: inactivity
Значение свойства — признака контроля	• true — если требуется проверять пропуска, для владельцев которого в поле **Контроль пропусков** переключатель установлен в положение **Включено**. • false — если требуется проверять пропуска, для владельцев которого в поле **Контроль пропусков** переключатель установлен в положение **Выключено**. В нашем примере: true

Режим работы	Выберите из раскрывающегося списка требуемые действия с пропусками: требуется ли просто помещать пропуска в отдельную папку для принятия решения оператором о дальнейших действиях с пропуском, либо необходимо приостанавливать действие найденных пропусков и помещать в папку. • Перемещение устаревших пропусков в папку — просто помещать пропуска в отдельную папку для принятия решения оператором о дальнейших действиях с пропуском. Пропуск остаётся действительным. Доступ по нему возможен. • Приостановка и перемещение устаревших пропусков в папку — приостанавливать и помещать в отдельную папку. Пропуск не удаляется из контроллеров доступа, но доступ блокируется.
Максимальный период неактивности, дней	Укажите временной интервал в днях (количество дней), по истечению которого пропуск без событий доступа считается устаревшим.
Анализировать время изменения пропуска	Установите флаг в поле, если, помимо событий доступа при расчёте активности пропуска требуется учитывать не только события доступа, но и события изменения пропуска оператором бюро пропусков.
Идентификатор свойства-маркера контроля	Уникальный идентификатор пользовательского свойства пропуска, значение которого будет устанавливаться в true для пропусков, которые приостановлены данным скриптом. В нашем примере: PausedByScrypt

Алгоритм работы

a. Поиск владельцев, для которых в поле с указанным идентификатором задано указанное значение.
b. Поиск активных пропусков для найденных владельцев (если пропуск уже приостановлен или находится в папке устаревших пропусков операции с ним не выполняются).
c. Поиск времени «активизации пропуска»:
    i. Если флаг в поле **Анализировать время изменения пропуска** установлен, то выбирается наиболее позднее из времени последнего редактирования и времени последнего прохода по пропуску
    ii. Если флаг не установлен, выбирается время последнего прохода по пропуску.
d. Если «время активизации» раньше чем , пропуск дополнительно помещается в папку устаревших пропусков.
e. Если указан режим работы «Приостановка и перемещение устаревших пропусков в папку», все найденные пропуска бессрочно приостанавливаются. В поле **Приостановлен с** задаётся текущее время.

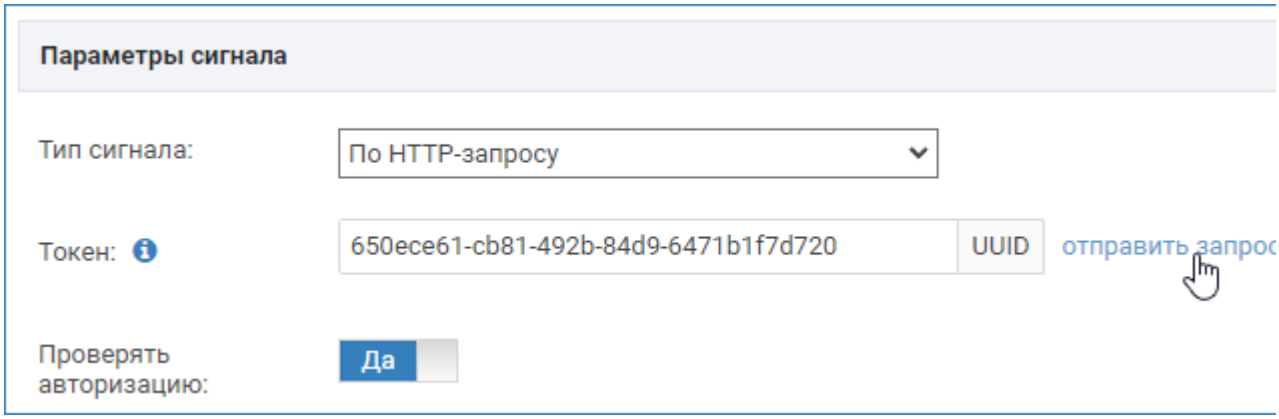
4. Нажмите на кнопку **Сохранить задание**.

ВАЖНО

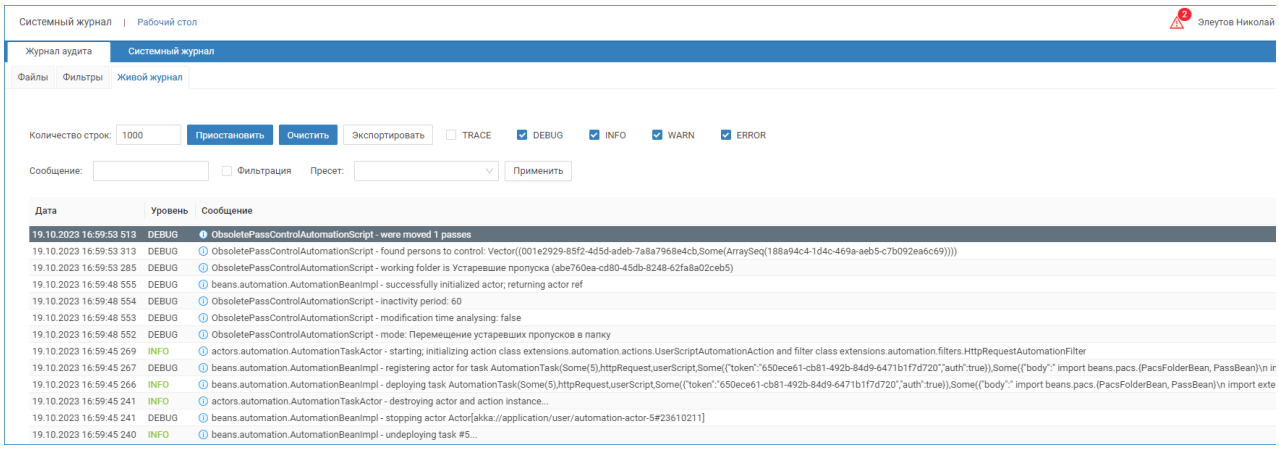
В НЕЙРОСС пропуск не «привязывается» жёстко к одной папке. Принадлежность пропуска папке пропусков является просто свойством пропуска. Пропуск может принадлежать нескольким папкам одновременно. При помещении пропуска в папку Устаревшие пропуска, пропуск остаётся также и в папке, в которой он располагался до запуска скрипта. Изъятие пропуска из папки осуществляется с помощью команды **Другие действия > Действие с папками > Изъять из текущей папки.**

Запуск задания

Дождитесь выполнения условия запуска задания или запустите задание вручную. Для ручного запуска в поле **Тип сигнала** выберите значение **По HTTP-запросу**, сохраните задание и нажмите отправить запрос.



Вы можете отслеживать процедуру инициализации и выполнения задания в Журнале аудита [Журнал аудита].



Запуск и выполненные скриптом действия фиксируются в системном журнале [Системный журнал].

Дата и время	Сообщение	Метки	Пользователь
19.10.2023 16:59:53 514	В папку 'Устаревшие пропуска' было перемещено 15 пропусков. Режим работы: Перемещение устаревших пропусков в папку. Анализировать время изменения пропуска: нет Максимальный период неактивности: 60 дней	Норма Автоматика	
19.10.2023 16:59:53 275	Запущен процесс контроля устаревших пропусков	Норма Автоматика	

Если в поле **Режим работы** задана необходимость приостановки пропуска. В системном журнале отражается факт блокировки действия на контроллерах и задания срока приостановки с текущего времени:

Дата и время	Сообщение	Метки	Пользователь
20.10.2023 12:26:39 869	Пропуск 'карта №175,429641, ПИН ?' обновлён: - состояние: «действителен» изменено на «приостановлен»	Изменение Пропуск	
20.10.2023 12:26:29 773	В папку 'Устаревшие пропуска' было перемещено 1 пропусков. Режим работы: Приостановка и перемещение устаревших пропусков в папку. Анализировать время изменения пропуска: нет Максимальный период неактивности: 60 дней	Норма Автоматика	
20.10.2023 12:26:29 762	Пропуск «карта №175/429641» обновлён	Пропуск Изменение	Автоматизаци
20.10.2023 12:26:29 641	Пропуск 'карта №175,429641, ПИН ?' обновлён - заблокирован, от: «20.10.23 12:26:29»	Ресурс Пропуск Изменение	Автоматизаци
20.10.2023 12:26:29 429	Запущен процесс контроля устаревших пропусков	Норма Автоматика	

Пропуска приостанавливаются с текущего времени, свойство **Приостановлен скриптом** у приостановленных пропусков устанавливается в значение **Да** (true).