ПРОГРАММНЫЙ ПРОДУКТ
СИСТЕМА УПРАВЛЯЮЩИХ ВОЗДЕЙСТВИЙ
INNOSTAGE ORCHESTRATOR

Руководство администратора

2022

АННОТАЦИЯ

Данный документ предназначен для специалистов, выполняющих администрирование программного продукта «Система управляющих воздействий **Innostage Orchestrator**» (далее - **Система**) и включает описание действий по администрированию **Системы**, по работе с программными коннекторами, сценариями и описание настройки интеграции с системами класса Incident Response Platforms и иными системами информационной безопасности.

СОДЕРЖАНИЕ

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение и функции Системы

Система предназначена для автоматизации типовых операций в области ИТ и ИБ, выполнения сценариев реагирования и воздействия на ИТ-инфраструктуру.

Система обеспечивает выполнение следующих функций:

- ведение справочников целевых ресурсов и хранение в защищённом виде учетных данных для доступа к ним;
- ведение программных коннекторов для подключения к целевым ресурсам;
- запуск сценариев реагирования и воздействия на ИТ-инфраструктуру;
- создание новых программных коннекторов и сценариев;
- интеграция с системами класса IRP и иными системами информационной безопасности в части автоматизации реагирования на типовые инциденты ИБ;
- ведение истории запусков сценариев.

1.2 Структура Системы

Система состоит из следующих основных компонентов:

- ПО «Система управляющих воздействий Innostage Orchestrator»;
- СУБД;
- Программные коннекторы.

Программное обеспечение, обеспечивающее выполнение функциональных возможностей **Системы**, и программные коннекторы, предназначенные для осуществления управляющих воздействий, устанавливаются на сервер приложений. СУБД, развернутая на сервере баз данных, осуществляет хранение конфигурационных параметров **Системы**, а также хранение информации по справочникам и сценариям.

Сервер приложений и сервер СУБД могут быть развернуты на одном физическом либо виртуальном сервере.

Управление **Системой** осуществляется посредством веб-консоли с АРМ администратора, работа с **Системой** осуществляется посредством веб-консоли с АРМ пользователей.

Структурная схема **Системы** представлена на рисунке ниже (Рисунок 1.1).

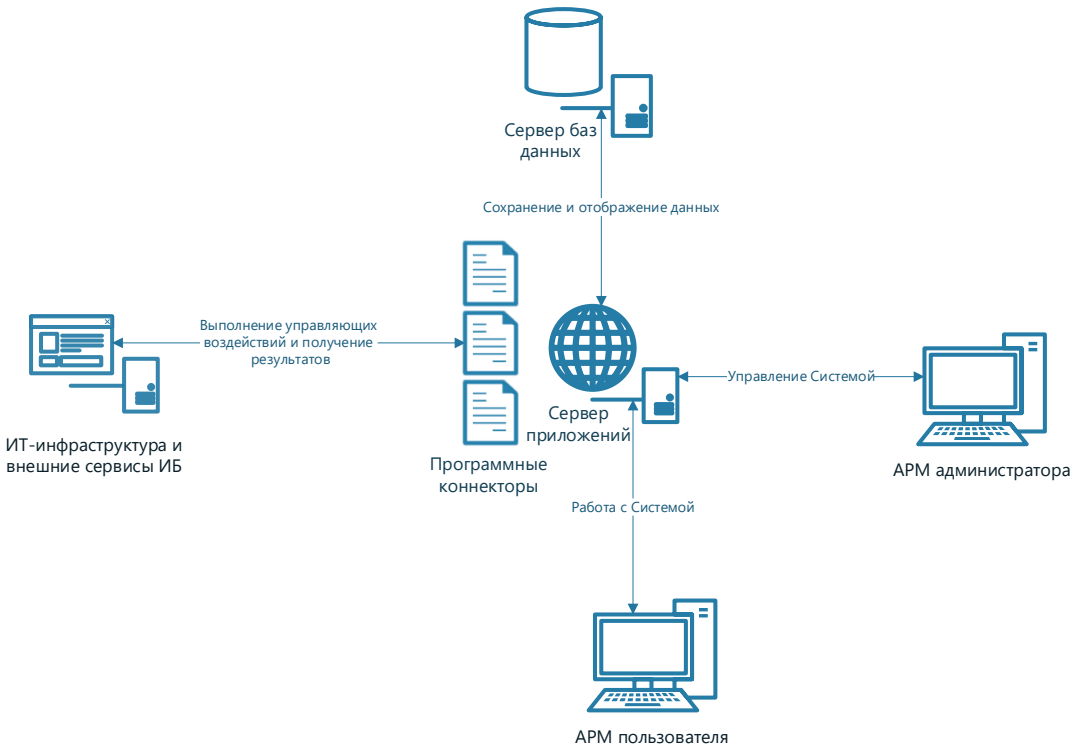

Рисунок 1.1 – Структурная схема

1.3 Требования к аппаратному и программному обеспечению

1.3.1 Требования к серверной части Системы

Все компоненты **Системы** устанавливаются на один виртуальный либо физический сервер под управлением операционной системы Astra Linux 2.12, РЕД ОС 7.3, Ubuntu 20.04.

Минимальные требования к аппаратной части:

- Процессор: не менее 2 ГГц, 4 ядра.
- Оперативная память: не менее 6 Гб.
- Дисковое пространство: не менее 100 Гб свободного дискового пространства.

Требования к программной части:

- ОС (64-разрядная): Astra Linux 2.12, РЕД ОС 7.3, Ubuntu 20.04.

– СУБД: Postgres Pro 11 и выше, PostgreSQL 11 и выше.
– Python 3.8.

1.3.2 Требования к рабочим станциям

Для работы с **Системой** рекомендуется использовать автоматизированное рабочее место со следующими аппаратными требованиями:

– Монитор с разрешением 1280x1024 и выше.
– Масштабирование дисплея 100%.

В качестве веб-обозревателя рекомендуется использовать актуальные версии браузера Google Chrome, Яндекс.Браузер, FireFox или Microsoft Edge.

1.4 Требования к персоналу, обеспечивающему функционирование Системы

1.4.1 Требования к квалификации администратора

Администратор **Системы** должен обладать квалификацией, обеспечивающей, как минимум:

– базовые навыки администрирования ОС семейства Linux (настройка репозиториев, системные настройки и т. д.);
– базовые навыки работы с СУБД PostgreSQL.

1.4.2 Требования к квалификации пользователя

Пользователь **Системы** должен обладать квалификацией, обеспечивающей, как минимум: базовые навыки работы на персональном компьютере с графическим пользовательским интерфейсом и базовые навыки работы с веб-обозревателями.

2 ОБЩИЕ ПРИНЦИПЫ ИСПОЛЬЗОВАНИЯ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

2.1 Вход в Систему

Для входа в **Систему** необходимо выполнить следующие действия:

1) На рабочей станции запустить веб-обозреватель.
2) В адресной строке веб-обозревателя указать URL адрес **Системы**.
3) В открывшейся форме (Рисунок 2.1) указать логин, пароль и осуществить вход в **Систему**, нажав на кнопку «**Войти**».

Рисунок 2.1 – Форма входа в Систему

2.2 Описание разделов меню

В результате успешной авторизации будет доступен графический интерфейс **Системы**.

На верхней панели располагается главное меню **Системы** (Рисунок 2.2), позволяющее получить доступ к основным функциям **Системы**:

- Сценарии – раздел, предоставляющий возможность работы со сценариями;
- Справочники – раздел, предоставляющий возможность работы с коннекторами, целевыми ресурсами, тегами и учетными данными;
- Пользователи – раздел, предоставляющий возможность работы с пользователями **Системы**;

– История запусков – раздел, предоставляющий возможность просмотра информации по запущенным сценариям;

– Интерактивный редактор – раздел, предоставляющий возможность создания новых программных коннекторов и сценариев;

– О системе – раздел, содержащий информацию о наименовании, версии, используемых библиотеках системы и информацию о контактах технической поддержки.

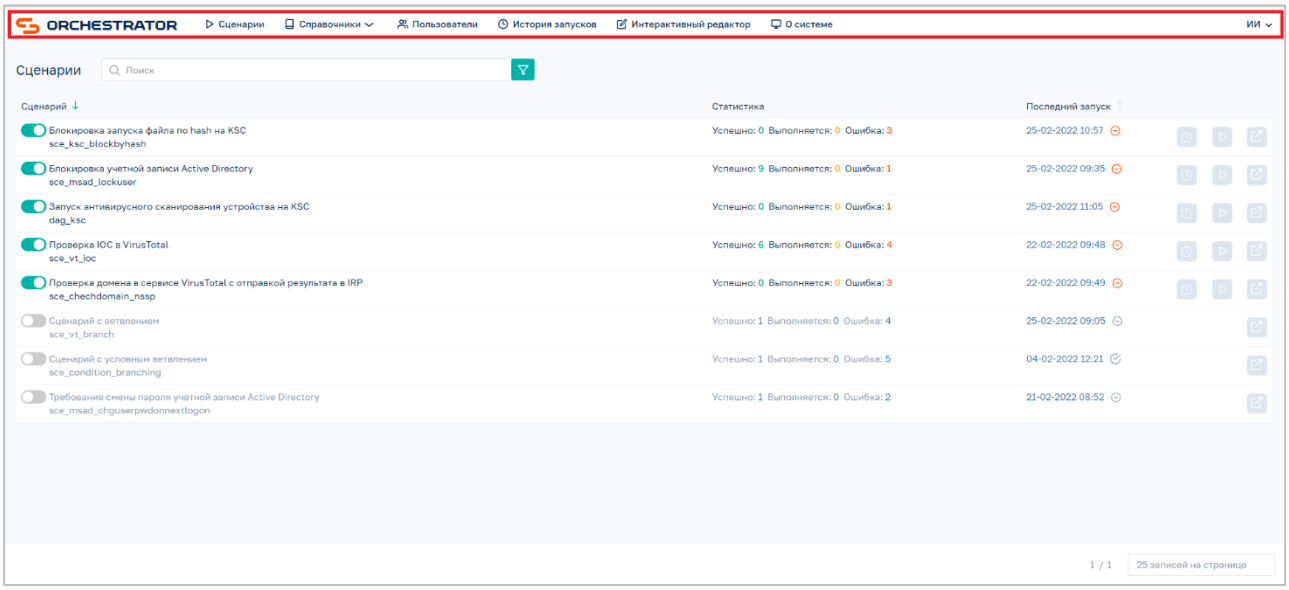

Рисунок 2.2 – Главное меню

2.3 Выход из Системы

Для выхода из **Системы** необходимо навести на имя пользователя в правом верхнем углу экранной формы и нажать на кнопку «**Выход**» (Рисунок 2.3).

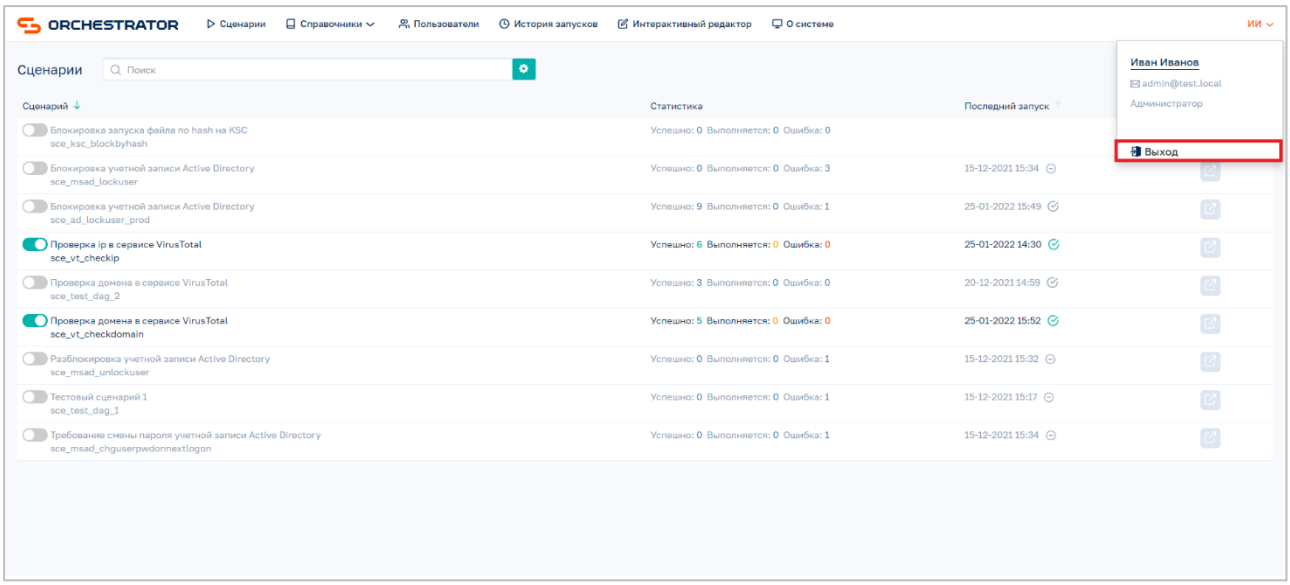

Рисунок 2.3 – Выход из Системы

3 АДМИНИСТРИРОВАНИЕ СИСТЕМЫ

3.1 Установка и обновление Системы

3.1.1 Установка Системы

Установка **Системы** описана в документе «Система управляющих воздействий INNOSTAGE ORCHESTRATOR. Руководство по развертыванию».

3.1.2 Обновление Системы

Для обновления необходимо расположить дистрибутивы обновления ПО на сервер в директорию /opt/orchestrator, где расположена **Система**, открыть терминал и выполнить следующие команды:

```
# Активация виртуального окружения
source /opt/orchestrator/venv/bin/activate

# Остановка сервисов
sudo service worker stop
sudo service scheduler stop
sudo service webserver stop

# Установка дистрибутива обновления
pip install -U /opt/orchestrator/nextstage-<version>-py3-none-any.whl

# Применение миграций
nextstage db upgrade
nextstage config migrate

# Проверка установленной версии системы
nextstage version

# Запуск сервисов
sudo service worker start
sudo service scheduler start
sudo service webserver start
```

3.2 Работа со службами Системы

Управление запуском и остановкой служб **Системы** осуществляется с помощью средств управления службами ОС (service, systemctl), на которой осуществлено развертывание.

Система включает в себя следующие службы:

- postgresql – служба СУБД PostgreSQL;
- redis-server / redis – служба очереди сообщений Redis;
- webserver – служба веб-сервера, обеспечивающая доступ к веб-интерфейсу Системы**;**
- scheduler – служба планировщика, отвечающая за постановку сценариев **Системы** на исполнение;
- worker – служба исполнителя задач **Системы**.

3.2.1 Запуск и остановка служб

Запуск и остановка служб **Системы** осуществляется с помощью выполнения команд в терминале ОС по следующему шаблону:

- при использовании service:
  sudo service <служба> <start | stop | restart>
- при использовании systemctl:
  sudo systemctl <start | stop | restart> <служба>

3.2.2 Мониторинг текущего состояния

Мониторинг текущего состояния служб **Системы** осуществляется с помощью выполнения команд в терминале ОС по следующему шаблону:

- при использовании service:
  sudo service <служба> status
- при использовании systemctl:
  sudo systemctl status <служба>

Текущее состояние работы можно отслеживать по статусу работы служб, в лог-файлах служб, а также в лог-файлах ПО **Системы**. Критерием текущего состояния также является доступность веб-интерфейса.

3.3 Резервное копирование данных Системы

3.3.1 Создание резервной копии

Для обеспечения восстановления данных **Системы** необходимо создание резервных копий БД. Для создания резервной копии БД необходимо выполнить следующие действия:

- остановить службы **Системы** (п.3.2.1);
- выполнить следующие команды:

```
# Переход в директорию расположения системы
cd /opt/orchestrator

# Формирование резервной копии БД: dumpname заменить на необходимое значение
sudo -u postgres pg_dumpall | gzip -c > dumpname.gz
```

- расположить резервную копию БД в хранилище;
- запустить службы **Системы** (п.3.2.1).

3.3.2 Восстановление из резервной копии

При возникновении критических ошибок, связанных с некорректной работой с СУБД, либо при утере важных данных необходимо выполнить восстановление резервной копии БД **Системы**. При восстановлении БД из резервной копии будут утеряны изменения, внесенные в **Систему** после создания резервной копии.

Для восстановления данных из резервной копии БД необходимо выполнить следующие действия:

- остановить службы **Системы** (п.3.2.1);
- разместить архив (.gz) резервной копии в директорию /opt/orchestrator;
- выполнить следующие команды:

```
# Переход в директорию расположения системы
cd /opt/orchestrator

# Распаковка архива резервной копии
gunzip dumpname.gz

# Восстановление БД из резервной копии: -d – наименование БД; -f – наименование дампа.
sudo -u postgres psql -d orchestrator -f dumpname
```

- запустить службы Системы (п.3.2.1).

3.4 Мониторинг и очистка лог-файлов Системы

В процессе работы **Система** осуществляет журналирование событий в лог-файлы, хранящиеся на сервере:

- лог-файлы служб **Системы**;
- лог-файлы работы сценариев **Системы**.

Хранение лог-файлов служб **Системы** осуществляется в следующих каталогах на сервере:

- каталог «/var/log/orchestrator»;
- каталог «/opt/orchestrator/logs/webserver»;
- каталог «/opt/orchestrator/logs/scheduler»;
- каталог «/opt/orchestrator/logs/worker».

Хранение лог-файлов работы сценариев **Системы** осуществляется в каталоге «/opt/orchestrator/logs/dag_runs». Очистка лог-файлов в данном каталоге не рекомендуется, т.к. приведет к утере технических логов процесса выполнения задач в рамках сценариев без возможности восстановления из резервной копии БД.

Рекомендуется периодическая очистка лог-файлов служб **Системы** для освобождения дискового пространства.

3.5 Настройка Системы

Для начала работы в **Системе** необходимо произвести ее настройку в следующем порядке:

- настроить подключение к базе данных;
- настроить подключение к службе очереди сообщений;
- настроить работу веб-сервера;
- настроить аутентификацию пользователей в **Системе**.

Настройка осуществляется с помощью интерфейса командной строки (CLI), предоставляемого **Системой**. Доступ к CLI осуществляется с помощью команды «nextstage». Для доступа к интерфейсу необходимо активировать виртуальное окружение:

```
# Активация виртуального окружения
source /opt/orchestrator/venv/bin/activate

# Вывод help информации по CLI
nextstage --help
```

Конфигурация осуществляется с помощью группы команд «config» интерфейса командной строки:

```
# Вывод help информации по CLI config
nextstage config --help
```

3.5.1 Настройка подключения к БД

Для настройки подключения к базе данных необходимо воспользоваться командой «db» группы команд «config»:

```
# Вывод help информации по команде config db
nextstage config db --help
```

С помощью данной команды можно осуществить как полную конфигурацию подключения к БД – вызвав команду без параметров, так и частичную – вызвав команду с параметрами. При частичной конфигурации будут установлены только указанные в параметрах значения, остальные значения конфигураций останутся без изменений.

Ниже представлено описание команды и ее возможных параметров для конфигурации подключения к БД:

Группа команд	**Команда**	**Возможные параметры**	**Описание параметров**
config	db	--db_name	Наименование БД
		--host	Адрес сервера БД
		--password	Пароль пользователя для подключения к БД
		--port	Порт подключения к серверу БД
		--username	Имя пользователя для подключения к БД

3.5.2 Настройка подключения к службе очереди сообщений

Для настройки необходимо воспользоваться командой «broker» группы команд «config»:

```
# Вывод help информации по команде config broker
nextstage config broker --help
```

С помощью данной команды можно осуществить как полную конфигурацию подключения к службе очереди сообщений – вызвав команду без параметров, так и частичную – вызвав команду с параметрами. При частичной конфигурации будут установлены только указанные в параметрах значения, остальные значения конфигураций останутся без изменений.

Ниже представлено описание команды и ее возможных параметров для конфигурации подключения к службе очереди сообщений:

Группа команд	**Команда**	**Возможные параметры**	**Описание параметров**
config	broker	--host	Адрес сервера службы очереди
		--login	Имя пользователя для подключения к службе очереди
		--password	Пароль пользователя для подключения к службе очереди
		--port	Порт подключения к серверу службы очереди

3.5.3 Настройка работы веб-сервера

Для настройки параметров работы веб-сервера необходимо воспользоваться командой «webserver» группы команд «config»:

```
# Вывод help информации по команде config webserver
nextstage config webserver --help
```

С помощью данной команды можно осуществить как полную настройку работы веб-сервера – вызвав команду без параметров, так и частичную – вызвав команду с параметрами. При частичной конфигурации будут установлены только указанные в параметрах значения, остальные значения конфигураций останутся без изменений.

Ниже представлено описание команды и ее возможных параметров для конфигурации работы веб-сервера:

Группа команд	Команда	Возможные параметры	Описание параметров
config	webserver	--host	Адрес работы веб-сервера
		--port	Порт работы веб-сервера
		--protocol	http / https
		--ssl_cert	Полный путь до файла сертификата для работы по протоколу https
		--ssl_key	Полный путь до файла ключа сертификата для работы по протоколу https
		--auth_type	Тип аутентификации в Системе: локальная (auth_db) / доменная (auth_ldap)
		--ldap_host	Адрес сервера LDAP
		--ldap_pass	Пароль пользователя для подключения к серверу LDAP
		--ldap_protocol	ldap / ldaps
		--ldap_search	Контекст поиска пользователей при аутентификации
		--ldap_uid_field	Атрибут имени пользователя LDAP
		--ldap_user	Пользователь для подключения к серверу LDAP (DN)
		--ldap_use_tls	Использование TLS при подключении

3.5.4 Настройка аутентификации

Система поддерживает два типа аутентификации пользователей:

- локальная аутентификация – аутентификация пользователей осуществляется с помощью БД **Системы**;
- доменная аутентификация – аутентификация пользователей осуществляется с помощью LDAP сервера.

Настройка типа аутентификации осуществляется в процессе настройки работы веб-сервера (п.3.5.3) и может быть осуществлена как в процессе полной настройки (команда без параметров), так и в процессе частичной конфигурации (с передачей необходимых параметров конфигурации).

Для настройки локальной аутентификации необходимо указать следующие параметры команды «webserver» группы команд «config»:

- auth_type = auth_db.

Для настройки доменной аутентификации необходимо указать следующие параметры команды «webserver» группы команд «config»:

- auth_type = auth_ldap;
- параметры команды, начинающиеся с префикса «ldap».

При изменении типа аутентификации с доменного на локальный, необходимо существующим пользователям **Системы** (через веб-интерфейс) задать пароль для возможности аутентификации.

3.6 Диагностика неисправностей

При недоступности веб-интерфейса **Системы** необходимо осуществить диагностику неисправностей в следующем порядке:

- осуществить проверку текущего состояния служб (п. 3.2.2);
- осуществить анализ лог-файлов служб на наличие ошибок (п.3.4);
- осуществить анализ лог-файлов ПО на наличие ошибок (п.3.4);

При обнаружении ошибок, влияющих на работоспособность **Системы**, произвести устранение неисправностей и при необходимости осуществить перезапуск служб с ошибочным статусом.

В случае, если неисправности не получается устранить самостоятельно, необходимо сформировать запрос в техническую поддержку программного продукта. К запросу приложить лог-файлы, скриншоты и описание выполняемых действий.

4 РАБОТА СО СЦЕНАРИЯМИ

Для работы со сценариями необходимо в главном меню выбрать раздел «Интерактивный редактор», на открывшейся странице в боковой панели слева выбрать раздел «Сценарии». Откроется перечень сценариев (Рисунок 4.1) со следующей информацией:

- Наименование сценария;
- Дата изменения;
- Размер.

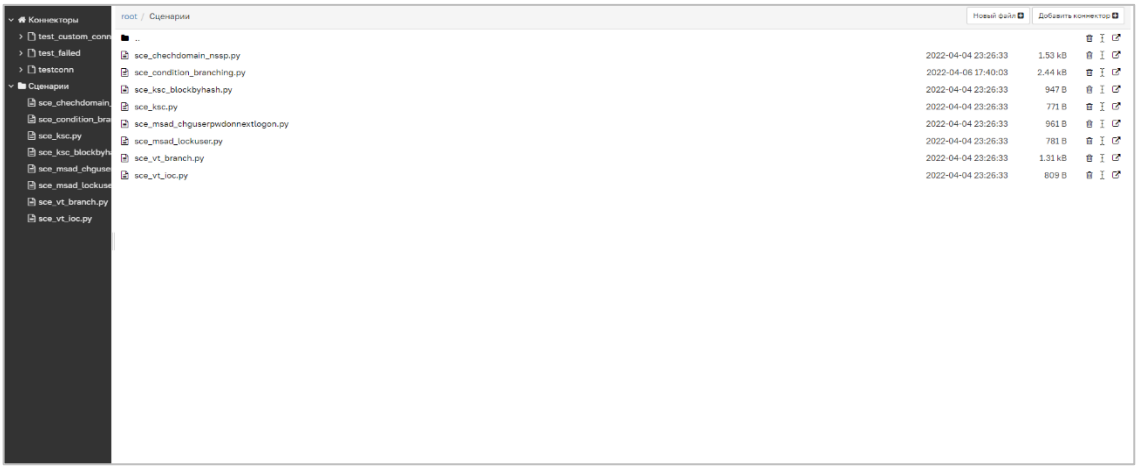

Рисунок 4.1 – Перечень сценариев

4.1 Создание сценария

Для создания сценария необходимо в перечне сценариев нажать на кнопку «**Новый файл**». Откроется форма создания с шаблоном кода сценария (Рисунок 4.2).

Шаблон кода сценария можно разделить на следующие блоки:

- блок импорта используемых модулей;
- блок параметров сценария;
- блок параметров задач сценария;
- блок последовательности выполнения задач.

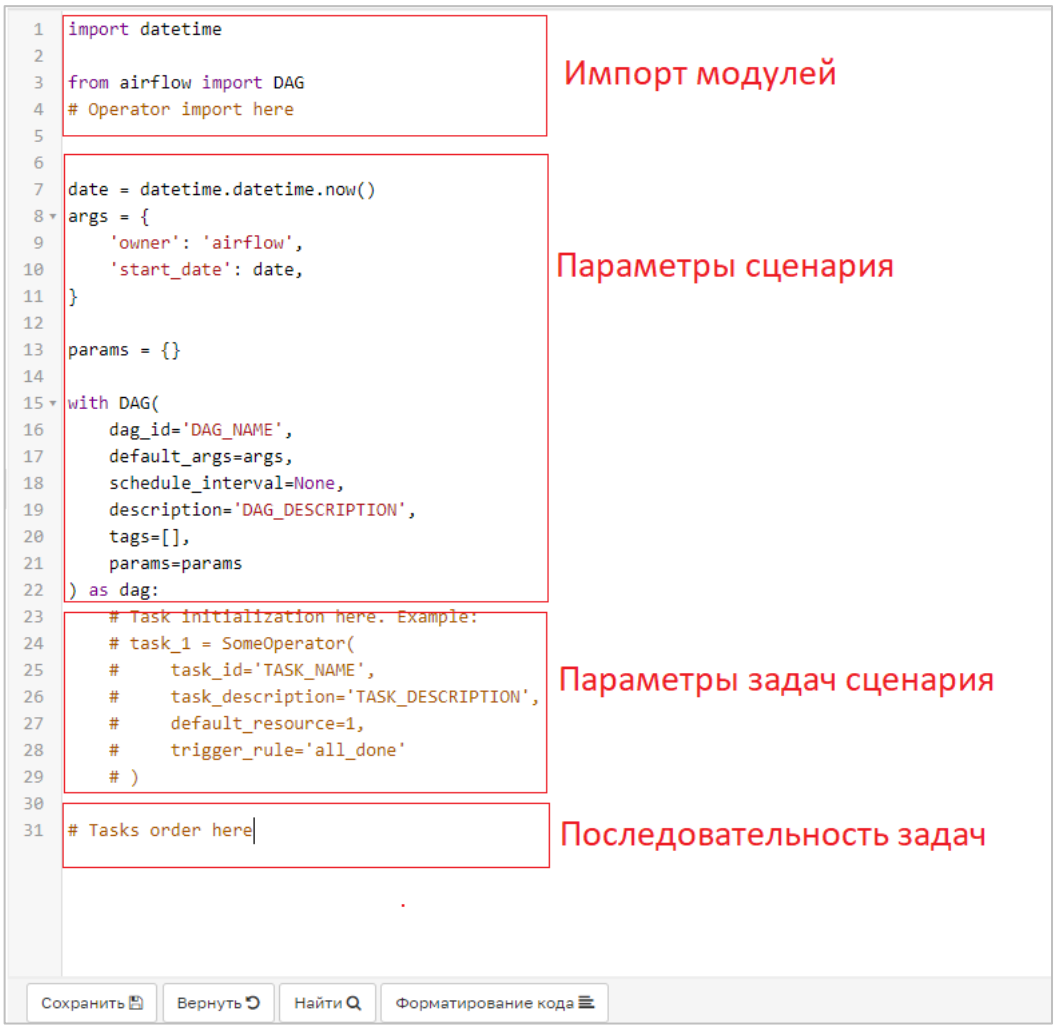

Рисунок 4.2 – Форма создания сценария

После внесения изменений на форму создания сценария необходимо нажать на кнопку «**Сохранить**». Откроется форма сохранения файла сценария, на которой необходимо ввести наименование файла в соответствии с шаблоном «<dag_id>.py» и сохранить сценарий (Рисунок 4.3).

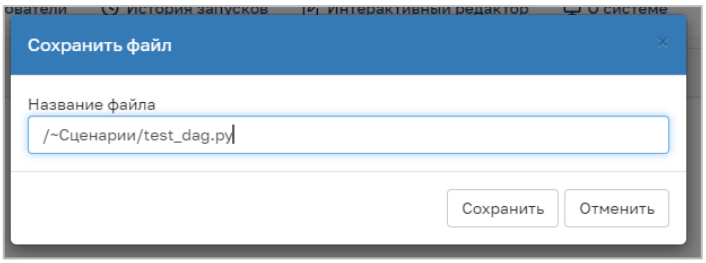

Рисунок 4.3 – Форма сохранения сценария

4.1.1 Блок импорта используемых модулей

Блок импорта используемых модулей необходим для добавления возможности использования модулей Python (класса сценария, классов операций коннекторов, различных функций) в файл сценария. Данный блок должен содержать импорты следующих модулей:

- по умолчанию:

```
import datetime
from airflow import DAG
```

- импорты модулей используемых операций коннекторов для задач сценария.

Импортирование модулей используемых коннекторов зависит от типа коннектора и осуществляется по следующему шаблону:

- для преднастроенных коннекторов:

```
from <package>.operators.<operator_file> import <OperatorClass>
```

- для кастомных коннекторов:

```
from connectors.<package>.operators.<operator_file> import <OperatorClass>
```

Для использования в качестве задачи сценария операции «условия» (ветвления в зависимости от успешности статуса выполнения предыдущей задачи в рамках сценария) необходимо импортировать следующий модуль:

```
from nextstage.common.branch_operator import BranchOperator
```

4.1.2 Блок параметров сценария

Блок параметров сценария предназначен для конфигурации параметров, необходимых для работы сценария и содержит:

- информацию о параметрах сценария по умолчанию;
- информацию о входных данных для задач сценария при работе сценария по расписанию;
- информацию о самом сценарии.

Информация о параметрах сценария по умолчанию всегда должна содержать следующий код:

```
date = datetime.datetime.now() - datetime.timedelta(days=1)
args = {
   'owner': 'airflow',
   'start_date': date,
}
```

Информация о входных данных для задач сценария при работе сценария по расписанию в момент создания сценария должна всегда содержать следующий код:

```
params = {}
```

Дальнейшая конфигурация входных данных и расписания сценария осуществляется с помощью веб-интерфейса системы на странице «Сценарии».

Информация о самом сценарии содержится в блоке контекстного менеджера «with Dag(…)» и содержит следующие данные:

- dag_id – уникальный строковый идентификатор сценария;
- default_args – параметры сценария по умолчанию;
- schedule_interval – расписание выполнения сценария;
- description – описание сценария;
- tags – используемые сценарием коннекторы;
- params – данные для выполнения сценария по расписанию.

Информация о сценарии в момент создания сценария всегда должна содержать следующий код:

```
with DAG(
        dag_id='<dag_id>',
        default_args=args,
        schedule_interval=None,
        description='<описание_сценария>',
        tags=[],
        params=params
) as dag:
```

4.1.3 Блок параметров задач сценария

Блок параметров задач сценария предназначен для объявления и описания используемых в рамках сценария задач.

Для использования реализованных операций коннекторов в задачах сценария, необходимо импортировать необходимые модули в соответствии с п. 4.1.1.

Добавление задачи в сценарий заключается в объявлении переменной экземпляра класса оператора (операции коннектора), наименование которой должно совпадать со свойством task_id объявленного экземпляра класса.

Экземпляр класса оператора должен содержать следующие данные:

- task_id – уникальный строковый идентификатор задачи в рамках сценария;
- task_description – описание задачи;
- default_resource – идентификатор ресурса по умолчанию, для которого будет выполняться задача;
- trigger_rule – всегда «all_done».

При использовании в качестве задачи сценария операции «условия», экземпляр класса должен содержать следующие данные:

- task_id – уникальный строковый идентификатор задачи в рамках сценария;
- task_description – описание задачи;
- on_success – список идентификаторов задач (task_id) успешного ветвления;
- on_failed – список идентификаторов задач (task_id) ошибочного ветвления;
- trigger_rule – всегда «all_done».

4.1.4 Блок последовательности выполнения задач

Блок последовательности выполнения задач предназначен для конфигурации последовательности задач в рамках сценария.

Конфигурация последовательности задач осуществляется на основе переменных задач сценария – экземпляров классов операторов (п. 4.1.3) – и может быть осуществлена с помощью следующего синтаксиса:

- >> – последовательное выполнение задач;
- [] – параллельное выполнение задач.

Примеры построения последовательности задач сценария:

```
# Пример 1
task_1 >> [task_2, task_3] >> task_4

# Эквивалентная примеру 1 запись
task_1 >> [task_2, task_3]
task_2 >> task_4
task_3 >> task_4

# Пример 2
task_1 >> branch_task >> [task_2, task_3]
task_2 >> task_4
```

4.2 Изменение сценария

Для перехода к редактированию сценария необходимо выбрать соответствующую строку перечня. Откроется страница редактирования сценария (Рисунок 4.4).

Рисунок 4.4 – Форма редактирования сценария

После внесения изменений в сценарий в соответствии с описанием блоков из п. 4.1, необходимо нажать кнопку «**Сохранить**» для фиксации изменений.

5 РАБОТА С КАСТОМНЫМИ КОННЕКТОРАМИ

Архитектура коннектора включает в себя следующие структурные элементы:

- hook (хук) – структурный элемент коннектора, предназначенный для реализации логики подключения к целевому ресурсу и содержащий наименование, порт работы по умолчанию;
- operation (операция) – структурный элемент коннектора, предназначенный для реализации логики операции, которую может совершить коннектор;
- operator (оператор) – структурный элемент коннектора, предназначенный для регистрации и использования реализованной операции в **Системе** и содержащий необходимые описания.

Для работы с коннекторами необходимо в главном меню выбрать раздел «Интерактивный редактор», на открывшейся странице в боковой панели слева выбрать раздел «Коннекторы». Откроется перечень кастомных коннекторов (Рисунок 5.1).

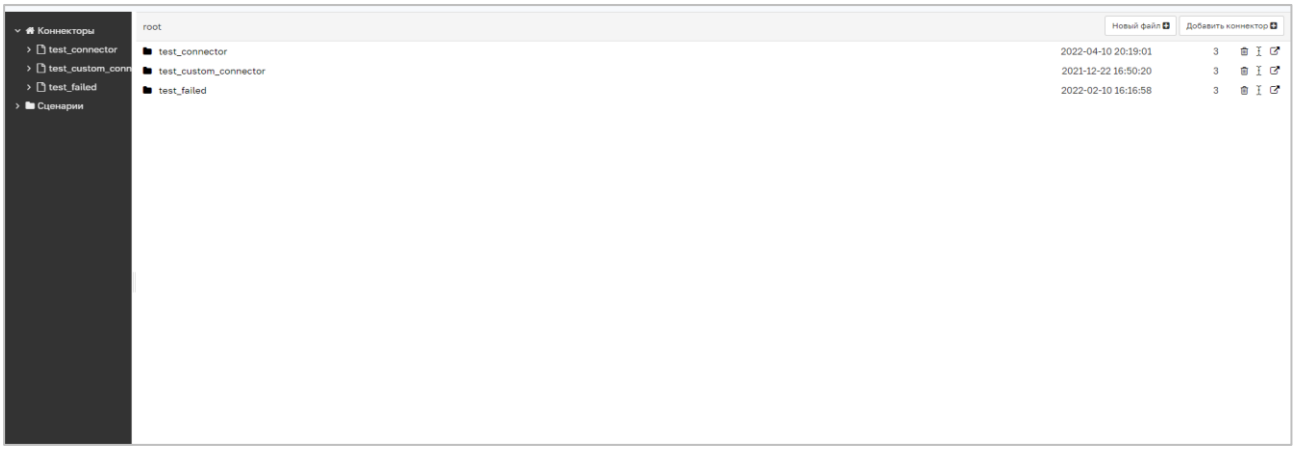

Рисунок 5.1 – Перечень коннекторов

5.1 Создание коннектора

Для создания коннектора необходимо в перечне коннекторов нажать на кнопку «**Добавить коннектор**». Откроется форма создания коннектора (Рисунок 5.2), в которую необходимо ввести наименование создаваемого коннектора (латиница) и нажать кнопку «**Добавить**».

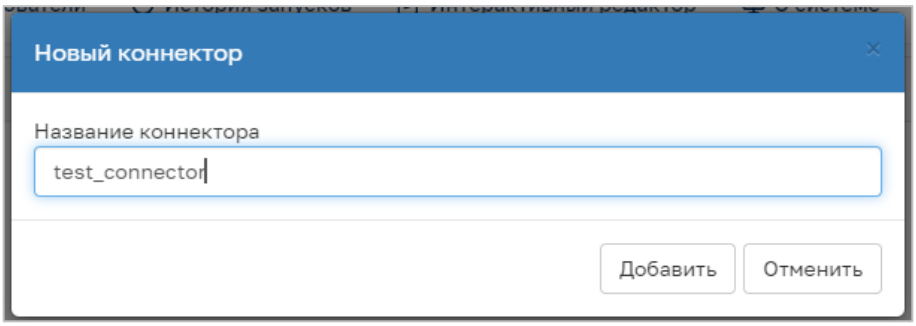

Рисунок 5.2 – Создание коннектора

После добавления коннектора произойдет создание нового коннектора по шаблону – будет создан каталог коннектора с необходимой структурой. В справочнике «Коннекторы» новый коннектор будет находиться в статусе «выполняется установка» до внесения и сохранения необходимых изменений. После внесения изменений статус коннектора будет изменен на успешный или ошибочный, в зависимости от корректности внесенных изменений.

Структура созданного коннектора представляет из себя каталог, содержащий следующие элементы:

- файл __init__.py – предназначен для хранения информации о версии коннектора;
- файл hook.py – предназначен для реализации хука коннектора;
- каталог operations – предназначен для реализации множества операций коннектора;
- каталог operators – предназначен для реализации множества операторов коннектора.

Реализация программного кода коннектора осуществляется в следующем порядке:

- реализация хука коннектора;
- реализация операции коннектора;
- реализация оператора коннектора.

5.1.1 Реализация хука

Реализация хука коннектора осуществляется в файле hook.py в классе, наследованном от базового класса CustomHook:

```
from nextstage.common.custom_hook import CustomHook
from nextstage.common.exceptions import *
from nextstage.security.auth_types import TYPE_BASE, TYPE_CERT

class TemplateHook(CustomHook):
    connector = ''
    default_port = 0
    extra_properties = {}
    supported_auth_types = []

    def get_conn(self):
        pass

    def test(self):
        pass
```

Описание свойств и функций реализуемого класса представлено ниже:

Свойство/функция	Тип	Обязательность	Описание
connector	str	да	Наименование коннектора
default_port	int	да	Порт по умолчанию, используемый коннектором
extra_properties	dict	нет	Дополнительные атрибуты, необходимые для конфигурации и отображаемые на форме целевого ресурса.
supported_auth_types	list	да	Поддерживаемые типы аутентификации коннектора. Возможные значения: TYPE_BASE, TYPE_CERT.
get_conn(self)	function	да	Метод получения сессии подключения к ресурсу для дальнейшего использования в операции. Возвращает объект сессии подключения.
test(self)	function	да	Метод проверки подключения к ресурсу

После внесения изменений в файл hook.py необходимо сохранить внесенные изменения, нажав кнопку «**Сохранить**».

5.1.2 Реализация операции

Для реализации операции необходимо перейти в каталог operations коннектора и нажать кнопку «**Новый файл**». Откроется форма создания операции коннектора с шаблоном кода операции.

Реализация операции коннектора осуществляется в классе, наследованном от базового класса ConnectorOperationInterface:

```
from nextstage.common.connector_operation import ConnectorOperationInterface
from nextstage.common.exceptions import *
from nextstage.common.result import SuccessResult


class TemplateOperation(ConnectorOperationInterface):

    @staticmethod
    def execute(session, resource, values, log) -> SuccessResult:
        pass
```

Описание свойств реализуемого класса представлено ниже:

Свойство/функция	Тип	Обязательность	Описание
def execute(session, resource, values, log)	function	да	Метод реализации логики операции. Возвращает объект класса SuccessResult или исключение: типизированное (из модуля nextstage.common.exceptions) или нетипизированное.

После внесения изменений на форму создания операции, необходимо нажать на кнопку «**Сохранить**». Откроется форма сохранения файла операции, на которой необходимо ввести наименование файла в соответствии с шаблоном «<operation>.py» и сохранить операцию (Рисунок 5.3)

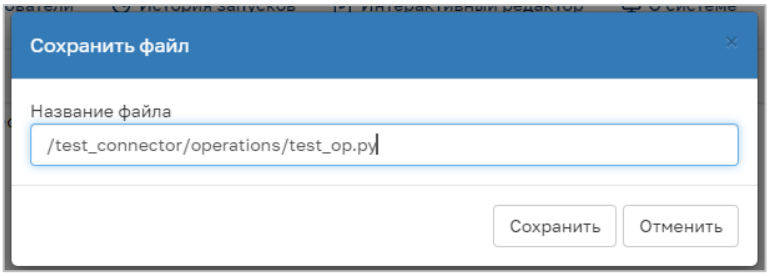

Рисунок 5.3 – Форма сохранения операции коннектора

5.1.3 Реализация оператора

Для реализации оператора необходимо перейти в каталог operators коннектора и нажать кнопку «**Новый файл**». Откроется форма создания оператора коннектора с шаблоном кода оператора.

Реализация оператора коннектора осуществляется в классе, наследованном от базового класса CustomOperator:

```
from nextstage.common.custom_operator import CustomOperator



class TemplateOperator(CustomOperator):
    operation_id = ""
    name = ""
    description = ""
    connector = TemplateHook
    operation = TemplateOperation
    input_values = {}
    output_values = {}
```

Описание свойств и функций реализуемого класса представлено ниже:

Свойство/функция	Тип	Обязательность	Описание
operation_id	str	да	Строковый идентификатор оператора, используемый по умолчанию
name	str	да	Наименование оператора
description	str	да	Описание оператора
connector	class	да	Класс реализованного хука, используемого оператором для подключения
operation	class	да	Класс реализованной операции, используемой оператором для выполнения логики
input_values	dict	нет	Входные параметры оператора, передаваемые в параметр values функции execute операции
output_values	dict	нет	Выходные параметры оператора

После внесения изменений на форму создания оператора, необходимо нажать на кнопку «**Сохранить**». Откроется форма сохранения файла оператора, на которой необходимо ввести наименование файла в соответствии с шаблоном «<operator>.py» и сохранить оператор (Рисунок 5.4).

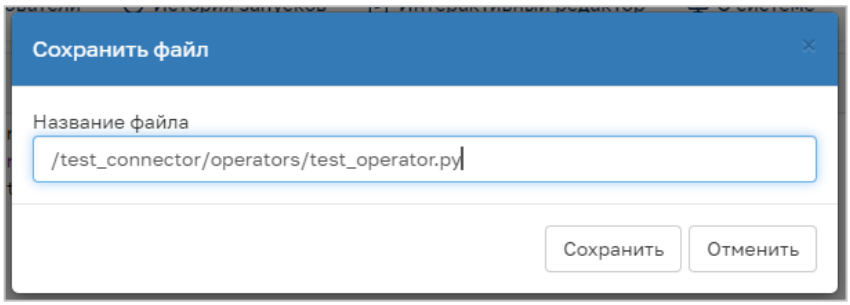

Рисунок 5.4 – Форма сохранения оператора коннектора

5.2 Изменение коннектора

Для перехода к редактированию коннектора необходимо выбрать соответствующую строку перечня. Откроется страница с файловой структурой коннектора (Рисунок 5.5).

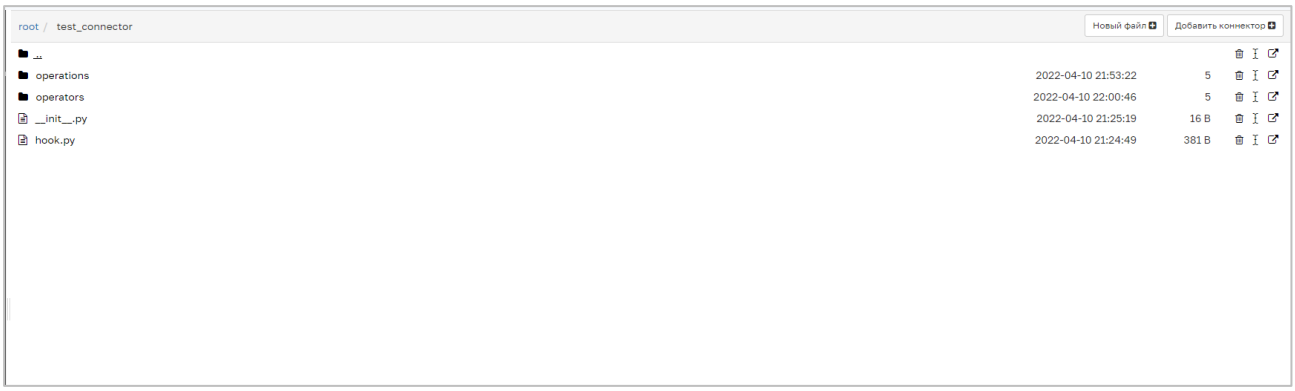

Рисунок 5.5 – Файловая структура коннектора

Редактирование коннектора осуществляется путем внесения изменений в реализацию необходимых структурных элементов коннектора, описанных в п. 5.1.

5.3 Установка дополнительных библиотек

При необходимости использования дополнительных пакетов (библиотек) языка программирования Python в реализации коннекторов, необходимо подключиться к серверу **Системы** и выполнить следующие команды:

```
# Активация виртуального окружения
source /opt/orchestrator/venv/bin/activate

# Установка дополнительных пакетов
pip install <package>
```

После установки дополнительных пакетов в виртуальное окружение **Системы**, данные пакеты будут доступны для использования в реализации структурных элементов коннектора.

5.4 Удаление коннектора

Для удаления коннектора необходимо в соответствующей строке перечня коннекторов нажать на кнопку «**Удалить**» и подтвердить действие в появившемся окне уведомления.

6 НАСТРОЙКА ИНТЕГРАЦИИ С ВНЕШНИМИ СИСТЕМАМИ

Интеграция с системами класса IRP и иными системами информационной безопасности в части автоматизации реагирования на типовые инциденты ИБ осуществляется с использованием программного интерфейса REST API. Описание поддерживаемых функций REST API представлено в веб-консоли «Swagger UI», которая доступна по умолчанию по адресу: http(s)://<адрес>:<порт>/orchestrator/api/ui.

В веб-консоли «Swagger UI» доступны следующие методы для интеграции с внешними системами:

- Получение списка доступных сценариев;
- Получение информации по сценарию;
- Получение результатов выполнения сценария;
- Запуск сценария;
- Получение статуса выполнения сценария.

Все действия в веб-консоли «Swagger UI» выполняются под учетной записью пользователя системы.

ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

Сокращение	Полное наименование
CLI	Command Line Interface
IRP	Incident Response Platform
АРМ	Автоматизированное рабочее место
БД	База данных
ИБ	Информационная безопасность
ИТ	Информационные технологии
ОС	Операционная система
ПО	Программное обеспечение
Система	Система управляющих воздействий Innostage Orchestrator
СУБД	Система управления базами данных

ПЕРЕЧЕНЬ ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ

Термин	Определение
Автоматизированное рабочее место	Программно-технический комплекс, предназначенный для автоматизации деятельности определенного вида
Коннектор	Программный модуль для взаимодействия Системы с целевым ресурсом.
Сценарий	Совокупность действий, выполняющихся в определенном порядке, предназначенных для выполнения управляющих воздействий на ИТ-инфраструктуру
Тег	Метка для группировки целевых ресурсов
Учетные данные	Совокупность данных, предназначенная для подключения к целевым ресурсам
Целевой ресурс	Средства защиты информации, компоненты ИТ-инфраструктуры, прикладные и автоматизированные системы, к которым Система имеет доступ для управления с помощью коннекторов