ООО «АЛЬМА Сервисез Компани»

# Интеллектуальная система мониторинга состояния динамического оборудования (АСТРА СМС)


# Руководство по установке и настройке

На 38 листах

Версия: 1.0.6

Москва, 2024

# АННОТАЦИЯ

Настоящий документ содержит:

- описание действий при развертывании Интеллектуальной системой мониторинга состояния динамического оборудования;
- описание действий при установке и настройке Интеллектуальной системой мониторинга состояния динамического оборудования;
- описание действий при обновлении Интеллектуальной системой мониторинга состояния динамического оборудования.

# Содержание

# Термины и сокращения

В таблице ниже (Таблица 1) приведены сокращения, термины и их определения, которые используются в документе.

**Таблица 1 – Сокращения, термины и определения**

| Сокращения/Термины | Определения |
|---|---|
| Аварийная уставка | Настраиваемое пороговое значение (LL - нижняя граница, HH - верхняя граница), при превышении которого выполняется регистрация события в Системе. Оборудование может выйти из строя |
| АСТРА СМС, Система | Интеллектуальная система мониторинга состояния динамического оборудования |
| БД | База данных |
| ГПЗ | Газоперерабатывающий завод |
| ДНС | Дожимная насосная станция |
| Доверительный интервал (от англ. confidence interval) | Выверенный, безопасный, исторический интервал работы датчика при нормальном режиме работы оборудования<br>Расчетное значение:<br>Дов.интервал = [Верхняя граница] - [Номинальное значение]<br>Дов.интервал = [Номинальное значение] - [Нижняя граница] |
| Зеленый коридор | Исторический диапазон показаний датчиков, при которых оборудование работает в нормальном режиме.<br>Зеленый коридор имеет границы. Границы являются расчетными. Расчеты строятся на основе доверительного интервала и номинального значения:<br>- верхняя граница: [Номинальное значение] + [Доверительный интервал];<br>- нижняя граница: [Номинальное значение] – [Доверительный интервал] |
| КВД | Компрессор высокого давления |
| КНД | Компрессор низкого давления |
| КПД | Коэффициент полезного действия |
| МВЗ | (Место возникновения затрат) Специальный код, описывающий структуру, где, на каком именно месте установлено оборудование |
| МРП | Межремонтный период |
| Наработка | Продолжительность работы оборудования с начала запуска |
| Настраиваемый интервал (conf интервал) | Настраиваемый интервал отклонения датчика от номинального значения |

| Сокращения/Термины | Определения |
|---|---|
| Номинальное значение | Историческое значение параметра, определяемое его функциональным назначением и служащее началом отсчета отклонений |
| НПЗ | Нефтеперерабатывающий завод |
| НПО | Нефтепромысловое оборудование |
| НПС | Нефтеперекачивающая станция |
| Операционная система | Программное обеспечение, управляющее компьютерами (включая микроконтроллеры) и позволяющее запускать на них прикладные программы |
| ПНиВ | Подготовки нефти и воды |
| Пользователь | Сотрудник организации, использующий Систему для решения стоящих перед ним задач |
| Предупредительная уставка | Настраиваемое пороговое значение (L - нижняя граница, H - верхняя граница), при превышении которого выполняется регистрация события в Системе. Оборудование может работать в нормальном режиме |
| Рабочая область | Главная часть приложения, в которой отображается информация выбранной в верхнем меню вкладки и в которой пользователь выполняет необходимые действия |
| Событие | Зарегистрированное Системой сообщение, содержащее информацию о нежелательном отклонении от нормативных показателей работающего оборудования |
| ТО | (Техническое обслуживание) Комплекс операций по поддержанию работоспособности оборудования при его эксплуатации, при ожидании (если оборудование в резерве), хранении и транспортировки. |
| ТОиР | Техническое обслуживание и ремонт |
| Тренд | График изменения какого-либо показателя за указанный промежуток времени |
| Узел | Укрупненный унифицированный (нормализованный) узел машины (комплекса машин), обладающий полной взаимозаменяемостью, самостоятельно выполняющий отдельные функции |
| УКПГ | Установка комплексной подготовки газа |
| УПН | Установка подготовки нефти |
| УППГ | Установка предварительной (первичной) подготовки газа |
| УПСВ | Установка предварительного сброса воды |
| Уставка | Настраиваемое значение контролируемого параметра (L - нижняя граница, H - верхняя граница), при котором происходит регистрация события в Системе |
| Linux | Семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит |

| Сокращения/Термины | Определения |
|---|---|
| | и программ проекта GNU, и, возможно, другие компоненты |
| Windows | Семейств коммерческих проприетарных операционных систем корпорации Microsoft, ориентированных на управление с помощью графического интерфейса |

# 1. Введение

## 1.1. Область применения

АСТРА СМС может использоваться на любом ответственном производстве, где необходимо организовать мониторинг за работой динамического и статического оборудования высокой критичности, преимущественно это объекты Нефтегазодобычи (как на суше, так и на море) и Нефтегазопереработки.

Объекты Нефтегазодобычи — это в основном площадочные объекты, на которых сконцентрировано большое количество динамического оборудования, которое отвечает за прием, подготовку и транспортировку углеводородов (ДНС, УПСВ, УПН, УКПГ, УППГ, ГПЗ, терминалы, НПС и т.д.).

Объекты Нефтепереработки — это НПЗ, где сконцентрировано большое количество динамического оборудования, выполняющего функцию перекачки сырой/готовой продукции, получаемой на нефтеперерабатывающих заводах.

## 1.2. Краткое описание возможностей

АСТРА СМС обеспечивает:
- мониторинг состояния оборудования.
- оценку (краткосрочную и долгосрочную) состояния оборудования.
- прогнозирование параметров работы оборудования.
- визуальную сигнализацию об отклонении в работе оборудования для конкретного узла (или другой составной части оборудования).
- регистрацию событий, связанных с:
  - остановкой оборудования;
  - выходом за уставки.
- ведение аналитической работы по оборудованию:
  - просмотр исторических данных;
  - построения трендов и диаграмм;
- оценку качества произведенного ремонта;
- работу с ЗИП:
  - замена ЗИП на оборудовании при выполнении ремонта или технического обслуживания;
  - хранение информации об установленных ЗИП и их состоянии;
- подсчет количества отработанных часов с момента установки ЗИП.

## 1.3. Уровень подготовки пользователя

Пользователь должен обладать базовыми навыками владения:
- персональным компьютером на базе операционной системы Microsoft Windows или Linux;
- пакетом офисных приложений (Microsoft Office).

## 1.4. Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю

Для работы с Системой необходимо ознакомиться со следующими документами:

- Руководство пользователя;
- Руководство администратора.

# 2. Назначение и условия применения

## 2.1. Виды деятельности, функции, для автоматизации которых предназначена Система

Система предназначена для:

- оценки (краткосрочной и долгосрочной) состояния оборудования;
- прогнозирование параметров работы оборудования;
- выдачи визуальной сигнализации об отклонении в работе оборудования для конкретного узла (или другой составной части оборудования):
- регистрации событий, связанных:

  - остановкой оборудования;
  - выходом за уставки (предупредительную и/или аварийную);

- ведение аналитической работы по оборудованию:

  - просмотр исторических данных;
  - построения трендов и диаграмм;

- оценки качества произведенного ремонта;
- оценки затрат на выполнение ремонта;
- представления информации пользователям Системы в соответствии с их уровнем доступа.

## 2.2. Условия применения Системы в соответствии с назначением

Для обеспечения корректной работы Системы требуется следующее программное обеспечение рабочего места пользователя:

1. Операционная система:

   - Windows 10 и старше.

2. Один из web-браузеров:

   - Microsoft Edge версии 84 и старше;
   - Google Chrome версии 84 и старше;
   - Mozilla Firefox версии 63 и старше.

**Примечание.** АСТРА СМС может работать в ОС Linux, если такая операционная система установлена на рабочем месте пользователя.

# 3. Установка и настройка АСТРА СМС

Для установки и настройки АСТРА СМС подготовлен дистрибутивный носитель с каталогами и файлами для установки.

Установка АСТРА СМС можно выполнить одним из следующих способов:

- 1-й способ — с помощью docker compose;
- 2-й способ — на физическую машину с операционной системой семейства Linux (нативная установка).

## 3.1. Состав дистрибутивного носителя

**Таблица 2 — Состав дистрибутивного носителя**

| Каталог | Файл | Назначение | Примечание |
|---|---|---|---|
| Backend (Серверная часть АСТРА СМС) | `AstraSMS-[Номер_релиза]rc[номер_релиз_кандидата]-cp[ver_Python]-cp[ver_Python]-linux_x86_64.whl` | Установочный пакет Python, где `[ver_Python]` — версия Python | В зависимости от версии Python `[ver_Python]` у Заказчика формируется установочный пакет с соответствующей версией Python `[ver_Python]` |
| | `docker.zip` | Установочный пакет docker файлов серверной части | Состав архива docker.zip:<br>- каталог nginx;<br>Файлы:<br>• `env.example` — файл переменных среды;<br>• `Dockerfile` — файл с инструкциями для создания docker-образа, из которого будет запускаться docker-контейнер;<br>`Docker-compose.yml` — файл с инструкциями для разворачивания среды с |

| Каталог | Файл | Назначение | Примечание |
|---|---|---|---|
| | | | несколькими Docker-контейнерами |
| | read.me | Файл с инструкцией по установке и настройке | |
| | server.zip | Установочный пакет серверной части АСТРА СМС | Содержит nginx, PosgreSQL, InfluxBD, Redis, Django |
| ML (Сервис расчета показателей состояния оборудования) | Astra_AI-[Номер_релиза]rc[номер_релиз_кандидата]-cp[ver_Python]-cp[ver_Python]-linux_x86_64.whl | Установочный пакет Python, где [ver_Python] – версия Python | В зависимости от версии Python [ver_Python] у Заказчика формируется установочный пакет с соответствующей версией Python [ver_Python] |
| | models_all_[X].zip | Установочный пакет всех методов расчетов, применяемых в ML для организации, где [X] – код организации, в которой будет выполняться установка | |
| | docker.zip | Установочный пакет docker файлов сервиса ML | |
| | read.me | Файл с инструкцией по установке и настройке | |
| Transfer Service (Сервис получения данных из внешних систем) | transfer_service-[Номер_релиза]rc[номер_релиз_кандидата]-cp[ver_Python]-cp[ver_Python]-linux_x86_64.whl | Установочный пакет Python, где [ver_Python] – версия Python] | В зависимости от версии Python [ver_Python] у Заказчика формируется установочный пакет с соответствующей версией Python [ver_Python] |
| | docker.zip | Установочный пакет docker файлов сервиса Transfer Service | |
| | read.me | Файл с инструкцией по установке и настройке | |
| Frontend | astra-sms-frontend-v[Номер_релиза]-*.zip | Установочный пакет клиентской части | |

| Каталог | Файл | Назначение | Примечание |
|---|---|---|---|
| Monitoring | monitoring.zip | Docker-образ с настройками мониторинга за работой Системы | |

## 3.2. Установка и настройка АСТРА СМС с помощью Docker Compose

### 3.2.1. Установка и настройка АСТРА СМС с помощью Docker Compose

Установка и настройка АСТРА СМС выполняется в несколько этапов:
1 этап — установка общесистемного программного обеспечения:

- операционная система;
- платформа Docker.

2 этап — создание сети Docker;
3 этап — установка и настройка серверной части АСТРА СМС (Backend), в том числе СУБД PostrgreSQL, СУБД InfluxDB, Django;
4 этап — установка и настройка сервиса ML для расчета показателей (astra-ai);
5 этап — установка и настройка Transfer Service;
6 этап — добавление Суперпользователя (с максимально возможными правами);
7 этап — установка и настройка клиентской части АСТРА СМС (Frontend);
8 этап — установка веб-сервера Nginx;
9 этап — импортирование подготовленных данных организации:

- структурные объекты организации;
- оборудование организации;
- составные части оборудования;
- датчики (сенсоры) оборудования и его составных узлов;

10 этап — создание пользователей в Системе.

#### 3.2.1.1. Установка общесистемного программного обеспечения

Установка операционной системы выполняется в соответствии с руководством по установке:

- Ubuntu — https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview;
- Astra Linux Common Edition — https://wiki.astralinux.ru/pages/viewpage.action?pageId=37290417.

Установка платформы Docker выполняется в соответствии с руководством по установке (https://docs.docker.com/engine/install/).

### 3.2.1.2. Создание сети Docker

Для взаимодействия сервисов необходимо создать сеть Docker, выполнив команду: `docker network create astra`.

### 3.2.1.3. Установка и настройка серверной части АСТРА СМС (Backend)

Для установки и настройки серверной части АСТРА СМС (backend) выполните следующие действия:

1. Создайте папку **backend**.
2. В созданный каталог сервиса АСТРА СМС скопируйте файл `license.yaml`.

**Примечание:** Файл лицензии выдается ответственному за установку сотруднику.

3. В созданную папку **backend** скопируйте файлы из каталога `backend` дистрибутива.
4. Извлеките из архива `docker.zip` файлы и каталоги в созданный каталог **backend**. В каталог **backend** добавлены из архива:

- каталог `nginx`;
- файлы:
  - `.env.example` — файл содержит набор переменных для настройки окружения.
  - `dockerfile` — файл с инструкциями для создания docker-образа, из которого будет запускаться docker-контейнер;
  - `docker-compose.yml` — файл с инструкциями для разворачивания среды с несколькими Docker-контейнерами.

5. Создайте копию файла `.env.example`.
6. Измените имя созданного файла на `.env`.
7. Откройте файл `.env` и выполните настройку (убрать знак #, если он установлен) нижеперечисленных параметров:

- Раздел ### Database ###:

```
POSTGRES_USER=astrasms
POSTGRES_PASSWORD=[пароль для пользователя astrasms]
POSTGRES_DB=astrasms
```

- Раздел ### Django project ###:

```
DEBUG=0
SECRET_KEY=[секретный ключ как можно сложнее] #не менее 32
символов
DJANGO_ALLOWED_HOSTS=*
CSRF_TRUSTED_ORIGINS=[адреса страниц, с которых выполняются
обращения к приложению, http://*.домен_организации.ru]
SQL_DATABASE=${POSTGRES_DB}
```

```
SQL_USER=${POSTGRES_USER} #должен совпадать с пользователем
POSTGRES_USER
SQL_PASSWORD=${POSTGRES_PASSWORD} #должен совпадать с
паролем POSTGRES_PASSWORD
SQL_HOST=postgres
SQL_PORT=5432
REDIS_HOST=redis
REDIS_PORT=6379
INFLUX_HOST=influxdb
INFLUX_PORT=8086
INFLUX_USER=astrasms
INFLUX_PASSWORD=password
INFLUX_DB=astrasms
DEFAULT_CACHE_LOCATION=redis://${REDIS_HOST}:${REDIS_PORT}
```

- Раздел ### Celery ###:

```
CELERY_BROKER_URL=redis://${REDIS_HOST}:${REDIS_PORT}
CELERY_BROKER_POOL_LIMIT=100
CELERY_RESULT_BACKEND=redis://${REDIS_HOST}:${REDIS_PORT}
CELERY_BACKEND_URL=redis://${REDIS_HOST}:${REDIS_PORT
```

- Раздел ### ML Digital Astra ###:

```
ASTRA_AI_URL=atra-ai #адрес сервиса ML
```

- Раздел ### Gunicorn ###:

```
GUNICORN_WORKER_CONNECTIONS=2
GUNICORN_WORKERS=1
```

- Раздел ### InfluxDB ###:

```
INFLUXDB_DB=${INFLUX_DB}
INFLUXDB_USER=${INFLUX_USER}
INFLUXDB_USER_PASSWORD=${INFLUX_PASSWORD}
INFLUXDB_DATA_QUERY_LOG_ENABLED=false
INFLUXDB_DATA_INDEX_VERSION=tsi1
```

8. В созданном каталоге **backend** выполните команду: `docker compose up -d`.

В результате выполненных действий собираются необходимые образы и запускаются docker-контейнеры со следующими компонентами:

- postrgesql;
- influxdb;
- redis;
- flower;
- celery-beat;
- celery-common-worker;
- celery-ml-worker;
- astra-sms.

### 3.2.1.4. Установка и настройка сервиса ML (astra-ai)

Для установки и настройки сервиса ML (astra-ai) выполните следующие действия:

1. Создайте папку **ML**.
2. В созданную папку **ML** скопируйте файлы из каталога `ML` дистрибутива.
3. В созданной папке **ML** создайте подкаталог **storage**.
4. Извлеките из архива `models_all_[X].zip` каталог `models` в созданный подкаталог **storage**.

**Примечание.** В дальнейшем в каталог `…/storage/models/` можно будет добавлять новые файлы для выполнения расчетов, не прибегая к перезагрузке самой Системы. Файлы должны быть написаны для версии Python, которая установлена.

5. Извлеките из архива `docker.zip` файлы и каталоги в каталог **ML**. В каталог **ML** добавлены файлы из архива:

   - файл `.env.example` — содержит набор переменных для настройки окружения;
   - `dockerfile` — файл с инструкциями для создания docker-образа, из которого будет запускаться docker-контейнер;
   - `docker-compose.yml` — файл с инструкциями для разворачивания среды с несколькими Docker-контейнерами;
   - `runserver.sh` — файл запуска сервиса расчетов ML.

6. Создайте копию файла `.env.example`.
7. Измените имя созданного файла на `.env`.
8. Откройте файл `.env` и выполните настройку (уберите знак #) нижеперечисленных параметров:

   - Раздел ### Docker Compose ###

```
# First you have to create a network, which you specify
here:
DOCKER_NETWORK_NAME=astra #сеть astra
# DOCKER_NETWORK_EXTERNAL=true
# Default tag for docker-image:
# DOCKER_ASTRA_AI_VERSION=latest
# External and internal API port:
# DOCKER_ASTRA_AI_EXTERNAL_PORT=8008 #закрытый порт.
Доступен только АСТРА СМС для выполнения расчетов
# DOCKER_ASTRA_AI_INTERNAL_PORT=8000
# The command for entry point to container, see
"runserver.sh".
# For production use Gunicorn (default):
# DOCKER_ASTRA_AI_COMMAND="gunicorn --bind 0.0.0.0:8000 --
preload"
# But for develop you can use Uvicorn with reloading:
# DOCKER_ASTRA_AI_COMMAND="uvicorn --host 0.0.0.0 --reload"
```

- Раздел ### ASTRA AI ###:

```
AI_MODELS_DIR=storage/models
# LOGGING_FILE=""
# LOGGING_LEVEL=INFO
# LOGGING_FORMAT="[%(asctime)s.%(msecs)03d] %(name)-14s
%(levelname)8s: %(message)s
%(module)s:%(funcName)s:%(lineno)s"
# LOGGING_DATEFMT="%Y-%m-%d %H:%M:%S"
# LOGGING_COLORED=yes
```

9. В созданном каталоге **ML** выполните команду: `docker compose up -d`.

В результате выполненных действий собирается образ astra_ai и запускается docker-контейнер astra_ai с сервисом astra_ai для расчета показателей здоровья оборудования и его составных элементов.

### 3.2.1.5. Установка и настройка Transfer Service

Для установки и настройки сервиса получения данных из внешних систем выполните следующие действия:

1. Создайте папку **Transfer Service**.
2. В созданную папку **Transfer Service** скопируйте файлы из каталога `Transfer Service` дистрибутива.
3. Извлеките из архива `docker.zip` файлы и каталоги в каталог **Transfer Service**. В каталог **Transfer Service** добавлены из архива:

- файлы:
  - `.env.example` — файл содержит набор переменных для настройки окружения.
  - `dockerfile` — файл с инструкциями для создания docker-образа, из которого будет запускаться docker-контейнер;
  - `docker-compose.yml` — файл с инструкциями для разворачивания среды с несколькими Docker-контейнерами.

4. Создайте копию файла `.env.example`.
5. Измените имя созданного файла на `.env`.
6. Откройте файл `.env` и выполните настройку (уберите знак #) нижеперечисленных параметров:

- Раздел ### Docker Compose ###:

```
# First you have to create a network, which you specify
here:
# DOCKER_NETWORK_NAME=astra # сеть astra
# DOCKER_NETWORK_EXTERNAL=true
# Default tag for docker-image:
# DOCKER_[ORG]_TRANSFER_VERSION=latest
# External and internal API port:
# DOCKER_[ORG]_TRANSFER_EXTERNAL_PORT=8009 # настройки порта
куда отправляются данные
```

```
# DOCKER_[ORG]_TRANSFER_INTERNAL_PORT=8000
```

- Раздел ### [ORG] TRANSFER ###

```
# HOST=localhost /*или 0.0.0.0*/
# PORT=8000
# GOOD_QUALITY_CODES="192,193,194,195,216,217,218,219" # при
необходимости изменить коды
INFLUXDB_HOST=influxdb # имя сервиса influxdb
# INFLUXDB_PORT=8086
# INFLUXDB_DATABASE=astrasms
# INFLUXDB_USERNAME="" # при необходимости
# INFLUXDB_PASSWORD="" # при необходимости
# INFLUXDB_TIMEOUT=""
# INFLUXDB_SSL=false

# LOGGING_FILE=""
# LOGGING_LEVEL=INFO
# LOGGING_FORMAT="[%(asctime)s.%(msecs)03d] %(name)-24s
%(levelname)8s: %(message)s
%(module)s:%(funcName)s:%(lineno)s"
# LOGGING_DATEFMT="%Y-%m-%d %H:%M:%S"
# LOGGING_COLORED=yes
```

7. В созданной папке **Transfer Service** выполнить команду: `docker compose up -d`.

В результате выполненных действий собирается образ transfer и запускается docker-контейнер transfer с сервисом Transfer Service для получения данных из внешних систем.

### 3.2.1.6. Добавление Суперпользователя (с максимально возможными правами)

Для создания Суперпользователя выполните команды в каталоге **Backend**, созданную в п. 3.2.1.3:

- создание Суперпользователя: `docker compose exec astra-sms python manage.py createsuperuser`;
- организация доступа до пространств в АСТРА СМС: `docker compose exec astra-sms python manage.py loaddata core_menusections users_roles events_reference accounting_equipmentgroups`.

### 3.2.1.7. Установка и настройка клиентской части

Для установки и настройки клиентской части выполните следующие действия:

1. Создайте новую папку **frontend**.
2. Извлеките из архива `astra-sms-frontend-v[номер_релиза]-*.zip` файлы и каталоги в созданный каталог **frontend**. В созданный каталог **frontend** добавлен каталог **build** с набором файлов:

- config.js — файл с настройками до хоста серверной части (backend);
- Изображения:
  - [имя_файла].gif;
  - [имя_файла].png;
  - [имя_файла].svg;
- index.html — главная страница для проверки выполнения работы приложения;
- Скрипты:
  - [имя_файла].js;
- [имя_файла].txt;
- [имя_файла].js.map;
- assets — каталог с иконками приложения.

### 3.2.1.8. Установка веб-сервера Nginx

Для запуска клиентской части АСТРА СМС требуется установит веб-сервер. В качестве веб-сервера используется Nginx.

Для установки веб-сервера Nginx выполните команду: `sudo apt install nginx`.

В каталоге установленного веб-сервера Nginx (`/etc/nginx/sites-enabled/`) создайте файл с конфигурацией `astrasms.conf`.

Пример файла конфигурации приведен ниже

```
server {
    listen 80;
    include /etc/nginx/mime.types;
    root /[path]/build; # путь до папки build, скопированной в созданный каталог Frontend

    location / {
        try_files $uri /index.html;
    }

    location /api {
        proxy_pass http://127.0.0.1:81/api;
        include proxy_params;
    }

    location /django/admin {
        proxy_pass http://127.0.0.1:81/django/admin;
        include proxy_params;
    }

    location /mediafiles {
        proxy_pass http://127.0.0.1:81/mediafiles;
        include proxy_params;
    }
```

```
    location /staticfiles {
        proxy_pass http://127.0.0.1:81/staticfiles;
        include proxy_params;
    }

    location /swagger-ui {
        proxy_pass http://127.0.0.1:81/swagger-ui;
        include proxy_params;
    }

    location /version {
        root /opt/astra-info/;
    }
}
```

В строке `root  /[path]/build;` вместо `[path]` укажите путь до каталога build с файлами пользовательского интерфейса АСТРА СМС, скопированного в каталог **frontend**.

Добавьте символьную ссылку с помощью команды: `sudo ln -s sites-available/astrasms.conf sites-enabled/`

Для проверки запуска приложения АСТРА СМС в адресной строке браузера укажите ip-адрес сервера, на котором развернуты все компоненты АСТРА СМС.

Откроется страница авторизации приложения.

### 3.2.1.9. Установка и настройка сервиса мониторинга

Для установки и настройки сервиса мониторинга выполните следующие действия:

1. Создайте папку Monitoring.
2. В папку Monitoring скопируйте файлы из каталога Monitoring дистрибутива.
3. Извлеките из архива `monitoring.zip` файлы и каталоги в каталог Monitoring. В каталог Monitoring добавлены из архива:

- docker-compose.yml;
- каталог log_monitor:

  • dockerfile — инструкции по сборке докер образа мониторинга
  • monitor.py — скрипт мониторинга логов
  • requirements.txt — список библиотек для скрипта мониторинга логов (monitor.py).

4. В папке Monitoring создать .env файл. В файле необходимо указать адрес электронной почты для отправки уведомлений об ошибках:

  • Раздел ####ORG####

```
NAME_ORG=[наименование организации, в которой установлена
Система]
```

- Раздел ####Email####

```
EMAIL_OUT=[адрес электронной почты, с которой отправляются
сообщения об ошибках]
EMAIL_IN_ORG=[адрес электронной почты администратора
организации, в которой эксплуатируется Система]
EMAIL_IN_TS=[адрес электронной почты технической поддержки
организации-разработчика Системы]
```

- Раздел ####MONITORING PERIOD####

```
PERIOD=[период времени отправления уведомлений, в секундах]
```

5. Выполнить команду `docker-compose up -d`.

В результате выполненных действий создается docker-контейнер для мониторинга работы Системы.

Работоспособность мониторинга подтверждается получением тестового уведомления на указанные адреса электронной почты: *«Выполнена установка и настройка сервиса мониторинга АСТРА СМС в организации [Наименование_организации]».*

В дальнейшем оправка уведомлений должна выполняться в соответствии заданной настройкой PERIOD.

### 3.2.2. Импорт данных (если они имеются)

Для импортирования данных должны быть подготовлены файлы с данными формата *.yaml:

- `orgs.yaml` — плоский словарь одной ветки структуры организации;
- `equipments.yaml` — список оборудования для объекта;
- `equipment-groups.yaml` — список групп оборудования для системы;
- `nodes.yaml` — иерархический список узлов для оборудования;
- `sensors.yaml` — список датчиков расположенных на узле/подузел оборудования;

Все файлы должны находиться в одном каталоге `client-data`.

**Примечание.** Каталог с файлами формата *.yaml создается для одного цеха. Если в организации несколько цехов, потребуется создания нескольких каталогов для каждого отдельного цеха с набором соответствующих файлов формата *.yaml.

Примеры заполненных файлов приведены в приложении (Приложение 1).

Для импорта данных выполните команды:

- копирование подготовленных файлов формата *.yaml с данными из каталога во временный каталог АСТРА СМС:

```
docker compose cp /path-to-your/client-data astra-sms:/tmp/;
```

- импортирование информации из файлов формата *.yaml в АСТРА СМС:

```
docker compose exec astra-sms python manage.py
import_client_data /tmp/client-data -v 2;
```

- импортирование паспортных данных оборудования:

```
docker compose exec astra-sms fill_equipment_specs;
```

- создание элементов дашборда:

```
docker compose exec astra-sms make_dashboard_widgets;
```

- импортирование статистических данных оборудования (часы наработки):

```
docker compose exec astra-sms import_equipment_statistic
equipment-statistic.csv  # optional
```

**Примечание.** Необходимую справочную информацию по работе с вышеперечисленными командами можно получить с помощью команды --help. Например, `docker compose exec astra-sms fill_equipment_specs –help`.

## 3.3. Обновление версии АСТРА СМС с помощью Docker Compose

**Примечание.** Перед выполнением обновления рекомендуется сделать резервные копии ранее созданных каталогов и файлов, базы данных.

При обновлении версии АСТРА СМС выполните повторяющиеся шаги для каждого обновляемого модуля:

1. Обновление серверной части АСТРА СМС (Backend):

   • В папку **backend** скопируйте обновленные файлы из каталога `backend` дистрибутива с заменой уже имеющихся там файлов. если выдан обновленный файл лицензии (license.yaml) добавьте его в каталог `backend`

   • извлеките из архива `docker.zip` файлы и каталоги в каталог **backend** с заменой уже имеющихся там файлов.

   • Выполните команду: `docker compose up -d --build`.

2. Обновление сервиса ML (astra-ai):

   • В папку **ML** скопируйте обновленные файлы из каталога `ML` дистрибутива с заменой уже имеющихся там файлов.

   • Извлеките из архива `models_all_[X].zip` каталог `models` в созданный подкаталог **storage** с заменой уже имеющегося каталога и файлов в нем.

   • Извлеките из архива `docker.zip` файлы и каталоги в каталог **ML** с заменой уже имеющихся там файлов.

   • Выполните команду: `docker compose up -d --build`.

3. Обновление сервиса получения данных из внешних систем (Transfer Service):

- В созданную папку **Transfer Service** скопируйте файлы из каталога `Transfer Service` дистрибутива с заменой уже имеющихся там файлов.
- Извлеките из архива `docker.zip` файлы и каталоги в каталог **Transfer Service** с заменой уже имеющихся там файлов.
- Выполните команду: `docker compose up -d --build`.

4. Обновление клиентской части:

- Извлеките из архива `astra-sms-frontend-v[Номер_релиза]-*.zip` файлы и каталоги в каталог **frontend** с заменой уже имеющихся там файлов и каталогов.
- Перезапустите веб-сервер.

5. Обновление сервиса мониторинга:

- Извлеките из архива `monitoring-v[Номер_релиза].zip` файлы и каталоги в каталог **Monitoring** с заменой уже имеющихся там файлов и каталогов;
- Выполните команду: `docker compose up -d --build`.

## 3.4. Установка и настройка АСТРА СМС (нативная)

Нативная установка АСТРА СМС выполняется в несколько этапов:

1 этап — установка общесистемного программного обеспечения:

- операционная система;

2 этап — установка специального программного обеспечения:

- СУБД PostgreSQL
- СУБД InfluxBD
- СУБД Redis;
- Celery;
- веб-сервер nginx;

3 этап — установка и настройка АСТРА СМС:

- установка и настройка серверной части (backend);
- установка и настройка сервиса ML для расчета показателей (astra-ai);
- установка и настройка Transfer Service;

4 этап — установка и настройка клиентской части АСТРА СМС;
5 этап — импортирование подготовленных данных организации:

- структурные объекты организации;
- оборудование организации;
- составные части оборудования;
- датчики (сенсоры) оборудования и его составных узлов;

6 этап — создание пользователей в Системе.

### 3.4.1. Установка общесистемного программного обеспечения

Установка операционной системы выполняется в соответствии с руководством по установке:

- Ubuntu — https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview;
- Astra Linux Common Edition — https://wiki.astralinux.ru/pages/viewpage.action?pageId=37290417.

### 3.4.2. Установка специального программного обеспечения

#### 3.4.2.1. Установка СУБД PostgreSQL

Установка СУБД PostgreSQL выполняется в соответствии с руководством по установке — https://www.postgresql.org/docs/.

#### 3.4.2.2. Установка СУБД InfluxBD

Установка СУБД InfluxDB выполняется в соответствии с руководством по установке — https://docs.influxdata.com/influxdb/v1/introduction/install/.

**Примечание:** Требуется установка СУБД Influx версии 1.7.

#### 3.4.2.3. Установка СУБД Redis

Установка СУБД Redis выполняется в соответствии с руководством по установке — https://redis.io/docs/install/install-redis/.

#### 3.4.2.4. Установка Celery

Установка Celery выполняется в соответствии с руководством по установке — https://docs.celeryq.dev/en/stable/getting-started/first-steps-with-celery.html#installing-celery/.

#### 3.4.2.5. Установка веб-сервера nginx

Установка nginx выполняется в соответствии с руководством по установке — https://www.nginx.com/resources/wiki/start/topics/tutorials/install/.

### 3.4.3. Установка и настройка АСТРА СМС

#### 3.4.3.1. Установка и настройка серверной части АСТРА СМС (backend)

Для установки и настройки серверной части АСТРА СМС выполните следующие действия:

1. Создайте каталог для сервиса АСТРА СМС (например, .../**backend**).
2. Скопируйте все файлы в созданный каталог сервиса АСТРА СМС.
3. В созданный каталог сервиса АСТРА СМС скопируйте файл `license.yaml`.

**Примечание:** Файл лицензии выдается ответственному за установку сотруднику.

4. Извлеките из архива `server.zip` все файлы и каталоги в созданный каталог **backend**.
5. Внесите изменения в файл `…/server/settings.py` (или создайте в текущем каталоге `secretsettings.py` для переопределения базовых настроек).
6. Создайте виртуальное окружение Python и перейдите в него: `virtualenv venv && source venv/bin/activate`.
7. Установите пакет из дистрибутива: `pip install *.whl`.
8. Выполните следующие команды:

```
python manage.py migrate
python manage.py collectstatic
python manage.py createsuperuser
python manage.py loaddata core_menusections users_roles
events_reference accounting_equipmentgroups
```

9. Создайте системные сервисы для запуска `gunicorn, celery` и запустите их.

Примеры для Systemd:

- /etc/systemd/system/astra-sms-gunicorn.service

```
[Unit]
Description=Gunicorn service for Astra SMS.
After=network.target

[Service]
User=astra-sms
Group=www-data
WorkingDirectory=/home/astra-sms/www
ExecStart=/home/astra-sms/www/venv/bin/gunicorn
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
PrivateTmp=true

Restart=always
RestartSec=5
Nice=-10
OOMScoreAdjust=-500
IOSchedulingClass=realtime
LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
```

- /etc/systemd/system/astra-sms-celery-beat.service

```
[Unit]
```

```
Description=Celery Beat Worker for Astra SMS
After=network.target

[Service]
Type=simple
User=astra-sms
Group=www-data
WorkingDirectory=/home/astra-sms/www
ExecStart=/bin/sh -c '/home/astra-sms/www/venv/bin/celery -A
server beat \
  --pidfile=/home/astra-sms/run/celery-beat.pid \
  --logfile=/home/astra-sms/log/celery-beat.log \
  --loglevel=INFO'

[Install]
WantedBy=multi-user.target
```

- /etc/systemd/system/astra-sms-celery-worker.service

```
[Unit]
Description=Celery Worker for Astra SMS
After=network.target

[Service]
Type=forking
User=astra-sms
Group=www-data
WorkingDirectory=/home/astra-sms/www
ExecStart=/bin/sh -c '/home/astra-sms/www/venv/bin/celery
multi start w1 w2 \
  -A server worker \
  -n common@%h \
  --concurrency=4
  --pidfile=/home/astra-sms/run/celery-common_%n.pid \
  --logfile=/home/astra-sms/log/celery-common.log \
  --loglevel=INFO'
ExecStop=/bin/sh -c '/home/astra-sms/www/venv/bin/celery
multi stopwait w1 w2 \
  --pidfile=/home/astra-sms/run/celery-common_%n.pid'
ExecReload=/bin/sh -c '/home/astra-sms/www/venv/bin/celery
multi restart w1 w2 \
  -A server worker \
  --concurrency=4 \
  --pidfile=/home/astra-sms/run/celery-common_%n.pid \
  --logfile=/home/astra-sms/log/celery-common.log \
  --loglevel=INFO'

[Install]
WantedBy=multi-user.target
```

### 3.4.3.2. Установка и настройка сервиса ML (astra-ai)

Для установки и настройки сервиса ML (astra-ai) выполните следующие действия:

1. Создайте каталог для сервиса АСТРА СМС (например, .../**astra-ai**).
2. Скопируйте все файлы в созданный каталог сервиса АСТРА СМС.
3. Извлеките из архива `models_all_[X].zip` все файлы и каталоги (при его наличии).
4. Если в дистрибутиве отсутствует архив `models_all_[X].zip` создайте пустой каталог моделей `.../storage/models`.
5. Создайте виртуальное окружение Python и перейдите в него: `virtualenv venv && source venv/bin/activate`.
6. Установите пакет из дистрибутива: `pip install *.whl`.
7. Создайте копию файла `.env.example`.
8. Измените имя созданного файла на `config`.
9. Откройте файл `config` и проверьте наличие нижеперечисленных параметров:

- Раздел ### Docker Compose ###

```
# First you have to create a network, which you specify
here:
DOCKER_NETWORK_NAME=astra
# DOCKER_NETWORK_EXTERNAL=true
# Default tag for docker-image:
# DOCKER_ASTRA_AI_VERSION=latest
# External and internal API port:
# DOCKER_ASTRA_AI_EXTERNAL_PORT=8008 /*закрытый порт.
Доступен только АСТРА СМС для выполнения расчетов*/
# DOCKER_ASTRA_AI_INTERNAL_PORT=8000
# The command for entry point to container, see
"runserver.sh".
# For production use Gunicorn (default):
DOCKER_ASTRA_AI_COMMAND="gunicorn --bind 0.0.0.0:8000 --
preload"
# But for develop you can use Uvicorn with reloading:
# DOCKER_ASTRA_AI_COMMAND="uvicorn --host 0.0.0.0 --reload"
```

- Раздел ### ASTRA AI ###:

```
AI_MODELS_DIR=storage/models
# LOGGING_FILE=""
# LOGGING_LEVEL=INFO
# LOGGING_FORMAT="[%(asctime)s.%(msecs)03d] %(name)-14s
%(levelname)8s: %(message)s
%(module)s:%(funcName)s:%(lineno)s"
# LOGGING_DATEFMT="%Y-%m-%d %H:%M:%S"
# LOGGING_COLORED=yes
```

10. Создайте системный сервис для запуска `gunicorn` и запустите его.

Пример для **Systemd**:

```
/etc/systemd/system/astra-ai-gunicorn.service

[Unit]
Description=Gunicorn service for Astra AI.
After=network.target

[Service]
User=astra-sms
Group=www-data
WorkingDirectory=/home/astra-sms/www/ai
EnvironmentFile=/home/astra-sms/www/ai/config
ExecStart=/home/astra-sms/www/ai/venv/bin/gunicorn
astra_ai.api:app \
  -k uvicorn.workers.UvicornWorker \
  --bind 0.0.0.0:8000 \
  --preload
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
PrivateTmp=true

Restart=always
RestartSec=5
Nice=-10
OOMScoreAdjust=-500
IOSchedulingClass=realtime
LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
```

### 3.4.3.3. Установка и настройка Transfer Service

Для установки и настройки сервиса получения данных из внешних систем выполните следующие действия:

1. Создайте каталог для сервиса **Transfer Service** (например, .../**transfer**).
2. Скопируйте все файлы в созданный каталог.
3. Создайте виртуальное окружение Python и перейдите в него: `virtualenv venv && source venv/bin/activate.`
4. Установите пакет из дистрибутива: `pip install *.whl`.
5. Создайте копию файла `.env.example`.
6. Измените имя созданного файла на `config`.
7. Откройте файл `config` и выполните настройку (уберите знак #) нижеперечисленных параметров:

- Раздел ### Docker Compose ###:

```
# First you have to create a network, which you specify
here:
DOCKER_NETWORK_NAME=astra
```

```
# DOCKER_NETWORK_EXTERNAL=true
# Default tag for docker-image:
# DOCKER_[ORG]_TRANSFER_VERSION=latest
# External and internal API port:
# DOCKER_[ORG]_TRANSFER_EXTERNAL_PORT=8009
# DOCKER_[ORG]_TRANSFER_INTERNAL_PORT=8000
```

- Раздел ### [ORG] TRANSFER ###

```
# HOST=localhost /*или 0.0.0.0*/
# PORT=8000
# GOOD_QUALITY_CODES="192,193,194,195,216,217,218,219" /*при
необходимости изменить коды*/
INFLUXDB_HOST=localhost
# INFLUXDB_PORT=8086
INFLUXDB_DATABASE=astrasms
# INFLUXDB_USERNAME="" /*при необходимости*/
# INFLUXDB_PASSWORD="" /*при необходимости*/
# INFLUXDB_TIMEOUT=""
# INFLUXDB_SSL=false

# LOGGING_FILE=""
# LOGGING_LEVEL=INFO
# LOGGING_FORMAT="[%(asctime)s.%(msecs)03d] %(name)-24s
%(levelname)8s: %(message)s
%(module)s:%(funcName)s:%(lineno)s"
# LOGGING_DATEFMT="%Y-%m-%d %H:%M:%S"
# LOGGING_COLORED=yes
```

8. Создайте системный сервис `transfer-service` и запустите его.

Пример для **Systemd**:

```
/etc/systemd/system/taneco-transfer-aiohttp.service

[Unit]
Description=AIOHTTP service for Transfer Service.
After=network.target

[Service]
User=astra-sms
Group=www-data
WorkingDirectory=/home/astra-sms/www/transfer
EnvironmentFile=/home/astra-sms/www/transfer/config
ExecStart=/home/astra-sms/www/transfer/venv/bin/transfer-
worker
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
PrivateTmp=true

Restart=always
RestartSec=5
Nice=-10
```

```
OOMScoreAdjust=-500
IOSchedulingClass=realtime
LimitNOFILE=16384

[Install]
WantedBy=multi-user.target
```

### 3.4.3.4. Установка и настройка клиентской части

Для установки и настройки клиентской части выполните следующие действия:

1. Создайте новую папку **frontend**
2. Извлеките из архива `astra-sms-frontend-v[номер_релиза]-*.zip` файлы и каталоги в созданный каталог **frontend**. В созданный каталог **frontend** добавлен каталог **build** с набором файлов:

- `config.js` — файл с настройками до хоста серверной части (backend);
- Изображения:
  - `[имя_файла].gif`;
  - `[имя_файла].png`;
  - `[имя_файла].svg`;
- `index.html` — главная страница для проверки выполнения работы приложения;
- Скрипты:
  - `[имя_файла].js`;
- `[имя_файла].txt`;
- `[имя_файла].js.map`;
- `assets` — каталог с иконками приложения.

### 3.4.3.5. Настройка веб-сервера Nginx

В каталоге веб-сервера Nginx (`/etc/nginx/sites-enabled/`) создайте файл с конфигурацией `astrasms.conf`.

Пример файла конфигурации приведен ниже

```
server {
    listen 80;
    include /etc/nginx/mime.types;
    root /[path]/build; # путь до папки build, скопированной
в созданный каталог Frontend

    location / {
        try_files $uri /index.html;
    }

    location /api {
        proxy_pass http://127.0.0.1:81/api;
```

```
        include proxy_params;
    }

    location /django/admin {
        proxy_pass http://127.0.0.1:81/django/admin;
        include proxy_params;
    }

    location /mediafiles {
        proxy_pass http://127.0.0.1:81/mediafiles;
        include proxy_params;
    }

    location /staticfiles {
        proxy_pass http://127.0.0.1:81/staticfiles;
        include proxy_params;
    }

    location /swagger-ui {
        proxy_pass http://127.0.0.1:81/swagger-ui;
        include proxy_params;
    }

    location /version {
        root /opt/astra-info/;
    }
}
```

В строке `root /[path]/build;` вместо `[path]` укажите путь до каталога build с файлами пользовательского интерфейса АСТРА СМС, скопированного в каталог **frontend**.

Добавьте символьную ссылку с помощью команды: `sudo ln -s sites-available/astrasms.conf sites-enabled/`

Для проверки запуска приложения АСТРА СМС в адресной строке браузера укажите ip-адрес сервера, на котором развернуты все компоненты АСТРА СМС.

Откроется страница авторизации приложения.

### 3.4.4. Импорт данных (если они имеются)

Для импортирования данных должны быть подготовлены файлы с данными формата *.yaml:

- `orgs.yaml` — плоский словарь одной ветки структуры организации;
- `equipments.yaml` — список оборудования для объекта;
- `equipment-groups.yaml` — список групп оборудования для системы;
- `nodes.yaml` — иерархический список узлов для оборудования;

- `sensors.yaml` — список датчиков расположенных на узле/подузел оборудования;

Все файлы должны складываться в одном каталоге `client-data`.

**Примечание.** Каталог с файлами формата *.yaml создается для одного цеха. Если в организации несколько цехов, потребуется создания нескольких каталогов для каждого отдельного цеха с набором соответствующих файлов формата *.yaml.

Примеры заполненных файлов приведены в приложении (Приложение 1).

Выполните импорт данных оборудования (если имеются подготовленные данные) с помощью команды:

```
python manage.py import_client_data /path-to-your/client-data -v 2
```

# 4. Аварийные ситуации

В случае возникновения ошибки при выполнении каких-либо действий в Системе необходимо сообщить о ней в службу технической поддержки с предоставлением информации:

- роль пользователя, с которой выполнялись действия;
- на какой вкладке выполнялись действия;
- порядок произведенных действий, приведшей к возникновению ошибки (в том числе о вводимой в Систему информации, если ошибка произошла при её вводе);
- снимок экрана с ошибкой.

# Приложение 1 Примеры файлов формата *.yaml для импорта данных

## Приложение 1.1 Файл orgs.yaml

Файл `orgs.yaml` представляет собой плоский словарь одной ветки структуры организации, где ключом является регистронезависимое имя категории `astrasms.core.choices.OrgCategory`, а значением - название этой структуры.

Ключи `factory` и `factory_object` являются обязательными.

**Пример 1**:

```
industry: ГиД (геологоразведка и добыча)
business_unit: ООО "Головная организация"
spot: ООО "Дочерняя организация"
oilfield: Нефтяной пункт
factory: Месторождение
factory_object: Цех
```

**Пример 2**:

```
industry: Нефтеперерабатка
business_unit: АО "Организация"
factory: Нефтеперерабатывающий комплекс АО "Организация"
factory_object: Нефтеперерабатывающий комплекс АО
"Организация"
```

**Пример 3**:

```
factory: Нефтеналивной пункт
factory_object: Отгрузка
```

## Приложение 1.2 Файл equipments.yaml

Файл `equipments.yaml` содержит список оборудования для объекта. Пример одной единицы оборудования:

```
# Кодовое имя оборудования (обязательно).
- codename: 2711a
  # Дата и время загрузки в Astra SMS (опционально).
  joined: 2022-09-03T00:00:00Z
  # Идентификатор (число или строка), по которому
связываются узлы и
  # конфигурации моделей расчёта (обязательно).
  id: 1
  # Тип системы (обязательно):
  #   GAS - Газ;
  #   OIL - Нефть;
  #   WATER - Вода;
  #   REFRIGERANT - Хладагент;
  #   HEAT_TRANSFER_FLUID - Теплоноситель;
```

```
  #   INTAKE_WATER - Забортная вода;
  system_type: GAS
  # Название оборудования (обязательно).
  title: КВД 2711А
  # Тип оборудования (опционально).
  equipment_type: null
  # Класс оборудования (опционально).
  equipment_class: null
  # Код датчика включения оборудования (опционально).
  switch_on_sensor: null
```

Тип системы регистронезависимо сопоставляется по имени с `strasms.devices.choices.SystemType`. Незарегистрированное имя вызовет ошибку загрузки.

Тип оборудования регистронезависимо сопоставляется по имени с `astrasms.devices.choices.EquipmentType`. Незарегистрированное имя не вызовет ошибку загрузки и будет установлено в `NULL`.

Класс оборудования регистронезависимо сопоставляется по имени с `astrasms.devices.choices.EquipmentClass`. Незарегистрированное имя не вызовет ошибку загрузки и будет установлено в `NULL`.

## Приложение 1.3 Файл equipment-groups.yaml

Файл `equipments-groups.yaml` содержит список групп оборудования для системы. Файл является необязательным.

Пример:

```
# Наименование группы (обязательно).
- title: Оборудование ГКА
  # Перечень кодов оборудования (обязательно).
  equipments: [2711a, 2711b, 2711c, 2901a, 2901b]
- title: Оборудование ГКА, 1 ступень
  equipments: [2711a, 2711b, 2711c, 2901a, 2901b]
- title: Оборудование ГКА, 2 ступень
  equipments: [2711a, 2711b, 2711c, 2901a, 2901b]
- title: Насосы ПНиВ
  equipments: [2201a, 2201b, 2201c, 2202a, 2202b, 2202c]
- title: Насосы ППД
  equipments: [1701a, 1701b, 1701c, 1702a, 1702b, 1702c,
1702d]
- title: Теплоносители
  equipments: [5802a, 5802b]
- title: Насосы ХА
  equipments: [5701a, 5701b, 5701c, 5701d, 5701e]
```

## Приложение 1.4 Файл nodes.yaml

Файл `nodes.yaml` содержит иерархический список узлов для оборудования. Пример:

```
# Указатель на оборудование должен быть как минимум у
корневого узла.
- equipment_id: 1
  # Идентификатор (число или строка), по которому
связываются узлы и
  # конфигурации моделей расчёта (обязательно).
  id: 1
  # Категория узла (опционально).
  category: compressor
  # Наименование узла, которое будет отображено
(обязательно).
  title: Компрессор
  # Список кодов датчиков, которые должны отображаться в
этом узле (опционально).
  sensors: []
  # Код датчика включения узла (опционально).
  switch_on_sensor: null
  # Подузлы этого узла (опционально).
  nodes:
    - id: 2
      title: Приводная сторона
      nodes:
        - id: 3
          title: Опорный подшипник
          sensors:
            - 1029AIT351.PV
            - 1029AVXE169.PV
            - 1029AVYE170.PV
            - 1029ATE153.PV
    - id: 4
      title: Неприводная сторона
      nodes:
        - id: 5
          title: Опорный подшипник
          sensors:
            - 1029AIT351.PV
            - 1029ATE152.PV
            - 1029AVXE167.PV
            - 1029AVYE168.PV
```

Категория узла регистронезависимо сопоставляется по имени с `astrasms.devices.choices.NodeCategory`. Незарегистрированное имя не вызовет ошибку загрузки, а установит категорию в `NULL`.

Заметьте, что узлы сопоставляются по их наименованию, поэтому при повторном импорте данных следует проверить не изменил ли кто-то в системе названия узлов.

## Приложение 1.5 Файл sensors.yaml

Файл `sensor.yaml` содержит список датчиков, расположенных на объекте. Пример:

```
# Код датчика, используемый для сопоставления с узлами и
конфигурациями
# моделей расчёта (обязательно).
- codename: 1017FT001.PV
  # Наименование датчика (обязательно).
  title: Регулирование минимальной производительности насоса
  # Категория датчика (опционально).
  category: FLOW
  # Признак того, что датчик используется для определения
включения узла или
  # оборудования (опционально).
  is_switch_on: false
  # Доверительный интервал датчика (опционально).
  confinterval: 200.45
  # Летний доверительный интервал датчика (опционально).
  summer: 200.45
  # Максимальное значение (опционально).
  maximum: 750.0
  # Единица измерения (опционально). Сопоставление
происходит по имени
  # `astrasms.core.choices.MeasureUnit` в верхнем регистре.
Не найденные в
  # перечне вызовут ошибку импорта.
  measure_unit: MQH
  # Минимальное значение (опционально).
  minimum: 202.0
  # Номинальное значение (опционально).
  nominal: 464.0
  # Список зон с перечнями значений (диаграмма) датчика
(опционально).
  diagram: []
```

Категория датчика регистронезависимо сопоставляется по имени с `astrasms.devices.choices.SensorCategory`. Незарегистрированное имя не вызовет ошибку загрузки, а установит категорию в `NULL`.

Единица измерения регистронезависимо сопоставляется по имени с `astrasms.core.choices.MeasureUnit`. Незарегистрированное имя вызовет ошибку загрузки.

## Приложение 1.6 Каталог calculations

Содержит файлы конфигурации моделей расчётов для узлов оборудования.

Имя файла является названием конфигурации, по которому её можно будет найти в базе данных.

Конфигурации могут быть двух типов:

- HEALTH_INDEX;
- EFFICIENCY_FACTOR.

Пример конфигурации модели расчёта **HEALTH_INDEX**:

```
# Идентификатор оборудования (обязательно).
equipment_id: 12
# Название модели расчёта в сервисе Astra AI (обязательно).
model: hi
# Идентификатор узла (обязательно).
node_id: 106
# Словарь параметров для модели (опционально).
params: {}
# Список датчиков с параметрами, по которым производится
расчёт (обязательно).
sensors:
# Код датчика (обязательно).
- codename: 10P1701A.ON
  # Информационная строка датчика (опционально).
  param: 'on'
  # Число веса датчика (опционально).
  weight: null
- codename: 1017FT001.PV
  weight: 1
- codename: 1017PDT004.PV
  weight: 1
- codename: 1017PZT003.PV
  weight: 0.01
# Тип конфигурации модели расчёта (обязательно).
type: HEALTH_INDEX
```

Пример конфигурации модели расчёта **EFFICIENCY_FACTOR**:

```
# Идентификатор оборудования (обязательно).
equipment_id: 3
# Название модели расчёта в сервисе Astra AI (обязательно).
model: filanovskogo.kpdcomp_predict_1
# Идентификатор узла, если он указан в файле списка узлов
(опционально).
node_id: null
# Словарь параметров для модели (опционально).
params:
```

```
  dp: 0.1
  k: 1.181
  minutes: 20160
# Признак того, что данная модель предсказывает значения КПД
на будущее и
# поэтому расчёт должен быть сохранён отдельно от основного
КПД (опционально).
prediction: 14days
# Список датчиков с параметрами, по которым производится
расчёт (обязательно).
sensors:
# Код датчика (обязательно).
- codename: 10ABUOSM.PV
  # Информационная строка датчика (опционально).
  param: 'on'
  # Число веса датчика (опционально).
  weight: null
- codename: 1027PT001.PV
  param: p1
- codename: 1027PT005.PV
  param: p2
- codename: 1027TT001.PV
  param: t1
- codename: 1027TT003.PV
  param: t2
# Ступень КПД, для которой предназначена данная конфигурация
(опционально).
stage: 1
# Тип конфигурации модели расчёта (обязательно).
type: EFFICIENCY_FACTOR
```