Разворачивание репозитория OSTree

1. Общие сведения

1.1 Назначение документа

Настоящий раздел устанавливает назначение и общий принцип работы скрипта /usr/bin/rpm-ostree-server/mkostreerepo (далее — mkostreerepo), предназначенного для создания и обновления **OSTree-репозитория НАЙС.ОС**. Документ ориентирован на администраторов и инженеров, выполняющих подготовку “каналов” поставки ОС (веток ref), а также на специалистов, обеспечивающих публикацию репозитория для потребителей (клиентов/инсталляторов/скриптов деплоя).

1.2 Назначение скрипта mkostreerepo

Скрипт mkostreerepo предназначен для выполнения регламентированного процесса: **создать** OSTree-репозиторий (если он ещё не инициализирован) либо **выполнить новый коммит** (обновить содержимое репозитория) на основе входного treefile JSON для rpm-ostree.

В рамках одного каталога репозитория скрипт обеспечивает: подготовку структуры данных репозитория (repo/), выделение кэша сборки (cache/), подготовку входного treefile (*.json), формирование (при необходимости) RPM репофайлов (*.repo), выполнение rpm-ostree compose tree и обновление ostree summary.

1.3 Область применения

- Подготовка и ведение **центрального OSTree-репозитория** НАЙС.ОС для стендов, ВМ-образов и “каналов” доставки ОС.
- Организация “веток” поставки (через ref) для релизов/минорных версий/сборочных профилей.
- Публикация данных репозитория по HTTP(S) для последующего потребления клиентскими процедурами (ostree pull, ostree admin deploy и т.п.).

1.4 Термины и сокращения (в контексте настоящего документа)

- **OSTree-репозиторий** — каталог данных, содержащий коммиты и метаданные поставки “атомарной” системы. В документе данные репозитория размещаются в ${REPOPATH}/repo.
- **summary** — агрегированная сводка репозитория, необходимая клиентам для обнаружения refs и ускорения pull; обновляется командой ostree summary --update.
- **ref** — идентификатор ветки/канала в OSTree (пример: niceos/5.2/x86_64/minimal); задаётся в treefile JSON полем "ref".
- **treefile JSON** — входной файл описания состава системы для rpm-ostree compose tree: источники репозиториев, список пакетов, units, параметры сборки.
- **REPODATAPATH** — путь к данным OSTree (${REPOPATH}/repo); **CACHEDATAPATH** — путь к кэшу compose (${REPOPATH}/cache).
- **.repo файлы** — RPM-репозитории источников пакетов (формат DNF/YUM), используемые на этапе compose для получения пакетов (например: niceos.repo, niceos-updates.repo, niceos-extras.repo).

1.5 Принцип работы: “инициализация” и “обновление”

Скрипт реализует два режима поведения в зависимости от состояния каталога данных репозитория:

- **Первичный запуск (репозиторий ещё не создан).** Если в каталоге ${REPODATAPATH} отсутствует структура refs (признак: нет каталога ${REPODATAPATH}/refs), выполняется инициализация OSTree-репозитория в режиме archive-z2, затем выполняется первый commit (compose).
- **Повторный запуск (репозиторий существует).** Если репозиторий уже инициализирован, выполняется очередной commit (compose) поверх существующего репозитория, после чего обновляется summary. Это обеспечивает наращивание истории коммитов и поддержку ref-веток.

1.6 Выходные данные (результат выполнения)

По результатам выполнения в каталоге REPOPATH формируются и/или

поддерживаются следующие элементы:

- repo/ — данные OSTree-репозитория (коммиты, refs, summary и служебные структуры);
- cache/ — кэш сборки для rpm-ostree compose;
- *.json — treefile (в частности, при отсутствии входного файла может быть сгенерирован niceos-base.json);
- *.repo — RPM репофайлы источников пакетов, если они не были предоставлены заранее и не задан режим кастомизации (-c).

Дополнительно выполняется обновление summary, что является обязательным условием для корректной работы клиентов, выполняющих pull из данного репозитория.

1.7 Принципиальные условия и ограничения

- Скрипт использует rpm-ostree и ostree. При отсутствии rpm-ostree предпринимается попытка установки через tdnf. Если tdnf отсутствует — требуется ручная установка зависимостей.
- Параметры запуска передаются в формате -r=/path, -p=/path/file.json (через знак =). Запуск без обязательного --repopath не допускается.
- При отсутствии входного JSON и отсутствии ${REPOPATH}/niceos-base.json скрипт **генерирует** минимальный treefile. Сгенерированный ref формируется как niceos/5.2/$(uname -m)/minimal; это следует учитывать при стандартизации веток.
- По умолчанию скрипт создаёт дефолтные *.repo файлы (если их нет) с baseurl=https://p3.niceos.ru/... и gpgcheck=1. Следует обеспечить наличие ключа RPM-GPG-KEY-NICEOS и корректную конфигурацию доверия.
- Режим -c/--customrepo предусматривает интерактивный запрос. В автоматизированных сценариях (CI) данный режим может быть неприемлем без предварительной подготовки репофайлов и исключения интерактива.

2. Подготовка окружения и запуск

2.1 Требования к системе

- Операционная система: НАЙС.ОС (рекомендуется выполнять на выделенном сборочном узле).
- Доступ к пакетному менеджеру: tdnf (используется для установки rpm-ostree,

если пакет отсутствует).
- Обязательные команды: rpm-ostree и ostree (скрипт проверяет наличие).
- Права: пользователь должен иметь право записи в каталог REPOPATH (обычно запуск выполняется от **root**).
- Ресурсы: место на диске под repo/ и cache/ (объём зависит от состава treefile и числа коммитов).

2.2 Параметры запуска и формат командной строки

Скрипт принимает параметры в форме ключ=значение. Использование пробельного формата (-r /path) в рамках данного скрипта не является основным и не рекомендуется. Обязательным является параметр -r/--repopath.

2.2.1 Опции

- **-r=... / --repopath=...** — путь к каталогу репозитория (обязательно).
- **-p=... / --jsonfile=...** — путь к treefile JSON (необязательно).
- **-c / --customrepo** — режим “кастомных репофайлов” (не генерировать дефолтные).
- **-h / --help** — справка по использованию.

2.3 Подготовка каталога репозитория (REPOPATH)

Скрипт гарантированно создаёт каталог репозитория и необходимые подкаталоги. При этом проверяется, что заданный путь не занят файлом. Далее путь нормализуется (realpath -m либо pwd -P), после чего создаются подкаталоги данных и кэша: ${REPOPATH}/repo и ${REPOPATH}/cache.

```
# пример: подготовка каталога под репозиторий
mkdir -p /srv/ostree/niceos
ls -la /srv/ostree/niceos
```

2.4 Подготовка treefile JSON (входные данные compose)

Скрипт поддерживает три сценария подготовки treefile: (1) входной JSON задан явно через -p=...; (2) входной JSON не задан, но в ${REPOPATH} уже существует niceos-

base.json; (3) входной JSON не задан и niceos-base.json отсутствует — в этом случае будет **сгенерирован минимальный** niceos-base.json.

2.4.1 Особенности сценария с -p=...

При задании -p=... входной JSON файл проверяется на существование и копируется внутрь каталога ${REPOPATH}. Далее скрипт использует копию, чтобы сборка была самодостаточной относительно выбранного каталога репозитория.

2.5 Подготовка RPM репофайлов (*.repo)

Для выполнения rpm-ostree compose tree необходим доступ к RPM источникам пакетов. По умолчанию (если не задан режим -c) скрипт создаёт дефолтные репофайлы niceos.repo, niceos-updates.repo, niceos-extras.repo в каталоге ${REPOPATH}, но только если такие файлы отсутствуют (без перезаписи).

В режиме -c/--customrepo предусмотрен интерактивный шаг: оператору предлагается подтвердить, что необходимые репофайлы уже созданы вручную в каталоге ${REPOPATH} и соответствуют списку "repos" внутри treefile JSON. Данный режим следует применять, когда источники пакетов должны отличаться от дефолтных.

2.5.1 Требование для gpgcheck=1

Дефолтные репофайлы создаются с gpgcheck=1 и ключом file:///etc/pki/rpm-gpg/RPM-GPG-KEY-NICEOS. Для успешной сборки необходимо обеспечить наличие ключа и корректную настройку доверия RPM. В противном случае compose может завершиться ошибкой проверки подписи.

2.6 Выполнение compose и формирование коммита

Основная операция выполняется командой rpm-ostree compose tree с параметрами: --unified-core, --cachedir (кэш), --repo (данные OSTree) и указанием treefile JSON. На время compose устанавливается маска прав umask 0022 (для предсказуемых прав доступа), по завершении маска возвращается к исходному значению.

```
# пример: логика compose (упрощённо, для понимания)
rpm-ostree compose tree --unified-core \
 --cachedir="/srv/ostree/niceos/cache" \
 --repo="/srv/ostree/niceos/repo" \
 "/srv/ostree/niceos/niceos-base.json"
```

```
# обязательное обновление summary для клиентов
ostree summary --repo="/srv/ostree/niceos/repo" --update
ostree summary -v --repo="/srv/ostree/niceos/repo"
```

2.7 Порядок запуска (типовые сценарии)

Ниже приведены типовые сценарии запуска, соответствующие стандартному режиму (генерация treefile/репофайлов при необходимости) и режиму использования заранее подготовленного JSON.

2.7.1 Сценарий A: создать репозиторий “с нуля” (autogen treefile и .repo при необходимости)

```
# создаст /srv/ostree/niceos (если отсутствует),
# создаст /srv/ostree/niceos/repo и /srv/ostree/niceos/cache,
# при отсутствии niceos-base.json — сгенерирует его,
# при отсутствии *.repo — создаст дефолтные,
# затем выполнит compose и обновит summary.
sudo /usr/bin/rpm-ostree-server/mkostreerepo \
 -r=/srv/ostree/niceos
```

2.7.2 Сценарий B: использовать заранее подготовленный treefile JSON

```
# входной JSON будет скопирован в каталог REPOPATH и использован оттуда
sudo /usr/bin/rpm-ostree-server/mkostreerepo \
 -r=/srv/ostree/niceos \
 -p=/home/stan/treefiles/niceos-minimal.json
```

2.8 Контроль кода завершения и первичная диагностика

При ошибке выполнения rpm-ostree compose tree скрипт завершает работу с кодом возврата, равным коду возврата compose. Это используется для интеграции с автоматизацией (CI) и регламентом контроля сборки. Первичный анализ выполняется по тексту ошибки compose и по проверке доступности RPM источников, ключей подписи и содержимого treefile.

2.9 Типовые причины отказов (на практике)

- Отсутствует tdnf, при этом rpm-ostree не установлен — требуется ручная

установка.
- Некорректный путь -p=... (файл не существует) — скрипт завершится ошибкой на проверке.
- Недоступны RPM источники из *.repo (ошибка сети, неверный baseurl, ограничения доступа).
- Включён gpgcheck=1, но отсутствует/не доверен ключ RPM-GPG-KEY-NICEOS — ошибки верификации подписи пакетов.
- Использован режим -c в неинтерактивном окружении — выполнение может “зависнуть” на запросе подтверждения.

3. Контроль результата, публикация и эксплуатация

3.1 Назначение этапа

Настоящий раздел устанавливает порядок контроля результатов работы mkostreerepo, а также требования к публикации OSTree-репозитория для потребителей. Контроль включает проверку структуры репозитория, наличия refs и summary, корректности ref из treefile, и проверку того, что репозиторий пригоден для клиентских операций pull/deploy.

3.2 Контроль структуры каталога репозитория

По завершении работы скрипта в REPOPATH должны присутствовать подкаталоги repo/ (данные OSTree) и cache/ (кэш compose). Наличие каталога repo/refs является признаком инициализированного репозитория и наличия ссылок на refs.

```
# пример: базовая проверка структуры
REPOPATH="/srv/ostree/niceos"
ls -la "${REPOPATH}"
ls -la "${REPOPATH}/repo" | head -n 50

# признак инициализации (refs должны существовать)
test -d "${REPOPATH}/repo/refs" && echo "OK: repo/refs присутствует" || echo "FAIL: repo/refs отсутствует"
```

3.3 Контроль refs и истории коммитов

Для подтверждения того, что compose создал коммит и записал ref, следует проверить список refs и историю по интересующему ref. Значение ref определяется treefile JSON полем "ref". В дефолтно генерируемом treefile ref имеет вид niceos/5.2/$(uname -m)/minimal.

```
# 1) посмотреть refs в репозитории
ostree refs --repo="/srv/ostree/niceos/repo" | sed -n '1,80p'

# 2) вывести историю коммитов по конкретному ref (пример)
ostree log --repo="/srv/ostree/niceos/repo" "niceos/5.2/$(uname -m)/minimal" | sed -n '1,120p'
```

3.3.1 Требование согласованности ref

Значение ref является **публичным идентификатором** канала поставки. Оно должно быть стандартизировано (релиз/архитектура/профиль), стабильно и документировано. Изменение ref без регламента приводит к “потере” канала для клиентов и усложняет автоматизацию.

3.4 Контроль summary (обязательное условие для клиентов)

После каждого commit скрипт выполняет обновление summary. Отсутствие актуального summary является типовой причиной проблем на стороне клиента при pull и обнаружении refs. Требуется подтвердить, что summary присутствует и обновляется.

```
# повторно обновить summary (при необходимости) и вывести подробную информацию
ostree summary --repo="/srv/ostree/niceos/repo" --update
ostree summary -v --repo="/srv/ostree/niceos/repo"

# косвенная проверка наличия summary-файлов (набор файлов может различаться по версии ostree)
find "/srv/ostree/niceos/repo" -maxdepth 2 -type f -name "summary*" -o -name "*.sig" | sed -n '1,80p'
```

3.4.1 Практическое правило

Для публикации репозитория “как сервиса” следует считать обновление summary обязательной частью процедуры. Клиентский pull должен быть возможен без ручных

действий на сервере после каждого выпуска/коммита.

3.5 Публикация репозитория по HTTP(S)

Для потребления репозитория клиентами требуется обеспечить доступность каталога данных ${REPOPATH}/repo по HTTP или HTTPS. Публикуется именно каталог repo, а не весь REPOPATH. Кэш cache/ является внутренним артефактом compose и не должен быть доступен потребителям.

```
# пример: (упрощенно) проверить, что каталог repo доступен на локальном узле
# (точный способ публикации зависит от Nginx/Apache/другого сервера)
ls -la /srv/ostree/niceos/repo | head -n 40

# пример: проверка доступности с клиента (должен возвращаться HTTP-ответ)
curl -I "http://REPO_SERVER/ostree/repo/" || true
```

3.5.1 Ограничение доступа и целостность

Репозиторий OSTree является точкой поставки системных образов. Публикация должна выполняться в контролируемом контуре: рекомендуется HTTPS или защищённая сеть/VPN, ограничение записи на каталог репозитория, а также регламент на обновления. Любая несанкционированная модификация данных репозитория приводит к невозможности гарантировать происхождение коммитов.

3.6 Проверка пригодности репозитория для клиента (контрольная процедура)

Ниже приведена контрольная процедура “на стороне клиента”, подтверждающая, что репозиторий доступен, ref существует, и клиент способен получить содержимое. Для теста достаточно временного каталога и отключения проверки подписи, если это допустимо в тестовом контуре. Для промышленного применения следует использовать штатную политику доверия.

```
# пример контрольной процедуры (на клиенте)
mkdir -p /tmp/ostree-test/repo
ostree --repo=/tmp/ostree-test/repo init --mode=archive-z2

# добавить remote (пример; gpg-verify в тесте можно отключить)
ostree remote add --repo=/tmp/ostree-test/repo --set=gpg-verify=false niceos
```

```
"http://REPO_SERVER/ostree/repo"

# получить refs
ostree remote refs --repo=/tmp/ostree-test/repo niceos | sed -n '1,80p'

# подтянуть конкретный ref (пример)
ostree pull --repo=/tmp/ostree-test/repo niceos "niceos/5.2/$(uname -m)/minimal"
```

3.6.1 Связь с развёртыванием системного образа

Клиентские процедуры развёртывания (в том числе скрипты подготовки дисковых образов) опираются на наличие опубликованного репозитория и корректного ref. Стабильность именования refs и доступность summary напрямую определяют предсказуемость развёртывания.

3.7 Диагностика и восстановление работоспособности

При отказах в compose или при проблемах у клиентов рекомендуется выполнять диагностику в следующем порядке: (1) проверить доступность RPM источников из *.repo; (2) проверить GPG ключи и политику доверия; (3) проверить корректность treefile JSON; (4) проверить актуальность summary; (5) при необходимости очистить кэш compose.

```
# 1) перегенерация summary (часто решает проблемы обнаружения refs)
ostree summary --repo="/srv/ostree/niceos/repo" --update
ostree summary -v --repo="/srv/ostree/niceos/repo"

# 2) очистка кэша compose (использовать при поврежденном/устаревшем кэше)
rm -rf "/srv/ostree/niceos/cache/"*
ls -la "/srv/ostree/niceos/cache"
```

3.8 Чек-лист контролёра (оперативная проверка за 5 минут)

- ${REPOPATH}/repo существует; ${REPOPATH}/repo/refs присутствует.
- Команда ostree refs --repo=... возвращает ожидаемые refs.
- Команда ostree summary -v --repo=... выполняется без ошибок.
- Каталог ${REPOPATH}/repo опубликован по HTTP(S) и доступен с клиента (проверка curl -I).
- Тестовый ostree pull рефа с клиента выполняется успешно.
- Политика доверия (GPG/HTTPS/ACL) соответствует требованиям контура

эксплуатации.