ИНТЕГРАЦИОННЫЙ ИНТЕРФЕЙС АГЕНТА

API v1 и протокол ROY

Продукт: СОПКА.Рой · Версия: 1.0.0 · Агент: 1.0.0 · Протокол: 1 · Редакция: 14.08.2026

Правообладатель ПО: Общество с ограниченной ответственностью «Каннам». ИНН 2543020645, ОГРН 1132543001502. Реквизиты приведены в отдельном разделе документации.

Назначение документа

Документ описывает прикладной HTTP API СОПКА.Рой версии 1: публичные проверки состояния, API агента, workspace API и служебные JSON-интерфейсы веб-консоли. Owner API в этот документ намеренно не включён, потому что он относится к отдельному административному контуру владельца инсталляции и не должен использоваться как интеграционный контракт агента или рабочей области.

Канонический внешний базовый путь сервера - `/roy`. Если reverse proxy публикует приложение без префикса, из приведённых URL удаляется первая часть `/roy`. Все временные метки передаются в UTC в формате ISO 8601.

Версии текущего контракта

Компонент	Версия	Комментарий
Windows-агент	`1.0.0`	Версия исполняемого клиента и пакета обновления.
ROY protocol	`1`	Транспортный контракт агента, подписи запросов и совместимость команд.
Agent HTTP API	`v1`	Префикс `/roy/api/v1/agent`.
Policy schema	`1`	Схема policy bundle, которую применяет агент.
Capabilities schema	`1`	Схема описания возможностей агента и runtime.

Версия программы агента не обязана совпадать с версией protocol/API. При несовместимой версии `protocol_version` или `api_version` сервер отвечает кодом `426` и возвращает поддерживаемые версии. Совместимые добавления полей выполняются в рамках v1.

Аутентификация и подпись

Этап	Заголовки	Назначение
Скачивание установщика	`X-COPKA-ROY-Registration-Key`	Проверяет право получить пакет агента до регистрации устройства.
Регистрация	`X-COPKA-ROY-Registration-Key`	Привязывает агент к рабочей области и выдаёт device token.
Рабочие запросы	`Authorization: Bearer ...`, `X-COPKA-ROY-Client-ID`	Проверяет токен устройства и соответствие client ID.
Подписанные запросы	`X-COPKA-ROY-Key-ID`, `Timestamp`, `Nonce`, `Sequence`, `Content-SHA256`, `Signature`	Защита от подмены тела запроса, повторов и устаревших запросов.

Подпись строится по строкам: версия подписи `ROY1` или `DLP1`, HTTP-метод, полный путь с query string, timestamp, nonce, sequence и SHA-256 тела. Сервер принимает timestamp только в небольшом временном окне, хранит nonce и требует монотонный sequence для каждого устройства.

```
ROY1
POST
/roy/api/v1/agent/heartbeat
2026-08-27T08:10:15Z
base64-or-uuid-nonce
42
0f343b0931126a20f133d67c2b018a3b...
```

Жизненный цикл агента

1. Установщик проверяет доступность сервера через `GET /roy/api/v1/agent/health`.
2. До регистрации установщик может получить пакет агента по регистрационному ключу.
3. Агент формирует локальный UUID, пару ключей подписи и вызывает `POST /roy/api/v1/agent/register`.
4. После регистрации сервер связывает client ID, workspace, public key и device token.
5. Агент отправляет heartbeat, получает команды, update offer и policy bundle при изменении версии.
6. События, снимки экрана, результаты поиска файлов и статусы команд отправляются отдельными endpoint.

7. Realtime-доставка команд выполняется через Socket.IO, а HTTP heartbeat остаётся резервным каналом.

Публичные проверки состояния

Метод	Маршрут	Применение
GET	`/roy/api/server_health`	Расширенная диагностика сервера для эксплуатации.
GET	`/roy/api/health/live`	Liveness probe контейнера или reverse proxy.
GET	`/roy/api/health/ready`	Readiness probe перед включением сервера в балансировку.
GET	`/roy/api/v1/agent/health`	Проверка API агента до регистрации и установки.

```
curl -s https://dlp.example.local/roy/api/v1/agent/health

{
  "status": "ok",
  "server_time": "2026-08-27T08:10:15Z",
  "base_path": "/roy",
  "agent_version": "1.0.0"
}
```

Agent API v1

Метод	Маршрут	Назначение
GET	`/roy/api/v1/agent/installer/package`	Скачать ZIP-пакет агента по регистрационному ключу или из веб-сессии администратора.
GET	`/roy/api/v1/agent/ping`	Проверить Bearer-токен, подпись запроса и совместимость версий.
POST	`/roy/api/v1/agent/register`	Зарегистрировать устройство или обновить identity при повторной установке.
POST	`/roy/api/v1/agent/heartbeat`	Обновить присутствие клиента, получить команды, политики и предложение обновления.
GET	`/roy/api/v1/agent/commands`	Получить ожидающие команды через HTTP fallback.
POST	`/roy/api/v1/agent/commands/<command_id>/ack`	Зафиксировать доставку, запуск, успех или ошибку команды.
GET	`/roy/api/v1/agent/policies/<client_id>`	Получить подписанный набор политик для конкретного клиента.
POST	`/roy/api/v1/agent/logs`	Пакетно загрузить события агента.
POST	`/roy/api/v1/agent/log`	Передать одиночное служебное событие агента.
POST	`/roy/api/v1/agent/media`	Передать media-событие как запись аудита.
POST	`/roy/api/v1/agent/browser_event`	Передать событие браузерного канала.
POST	`/roy/api/v1/agent/screenshot`	Загрузить PNG или JPEG снимок экрана multipart/form-data.
POST	`/roy/api/v1/agent/file_search/results`	Передать результаты или статусы задания поиска файлов.
POST	`/roy/api/v1/agent/file_search/preview_sessions/<id>/content`	Загрузить содержимое файла для предпросмотра.
POST	`/roy/api/v1/agent/file_search/download_sessions/<id>/content`	Загрузить содержимое файла для контролируемой выдачи пользователю.

Метод	Маршрут	Назначение
GET	`/roy/api/v1/agent/version`	Получить сведения о доступной версии агента.
GET	`/roy/api/v1/agent/download`	Скачать текущий пакет агента авторизованным устройством.
POST	`/roy/api/v1/agent/update/download/<grant_id>`	Скачать обновление по выданному grant и update key.
POST	`/roy/api/v1/agent/update/status`	Передать состояние обновления: downloaded, installing, updated или failed.
POST	`/roy/api/v1/agent/remote_shell/events`	Передать зашифрованные события remote shell: answer, output или status.

Регистрация агента

Регистрация выполняется до появления Bearer-токена. Клиент передаёт workspace, локальный идентификатор, сведения об ОС, версии протокола/API, список возможностей и публичный ключ для дальнейшей подписи запросов.

```
POST /roy/api/v1/agent/register
X-COPKA-ROY-Registration-Key: rgk_...
Content-Type: application/json

{
  "workspace_id": "01a04400-0000-7000-8000-000000000001",
  "client_uuid": "01a04411-2f4a-7b9e-9a31-2fb6b17e7810",
  "hostname": "WIN-OPS-042",
  "platform": "Windows 11 Pro",
  "os_family": "windows",
  "agent_version": "1.0.0",
  "protocol_version": 1,
  "api_version": 1,
  "capabilities": {
    "file_events": true,
    "clipboard": true,
    "screenshots": true,
    "file_search": true
  },
  "key_id": "copka_key_2b6f...",
  "public_key": "base64-ed25519-public-key"
}
```

```
{
  "status": "ok",
  "client_id": "01a04411-2f4a-7b9e-9a31-2fb6b17e7810",
  "device_token": "returned-on-registration",
  "protocol_accepted": true,
  "api_accepted": true,
  "protocol_version": 1,
  "api_version": 1,
  "policy_schema_version": 1
}
```

Heartbeat и команды

Heartbeat сообщает серверу, что агент online, обновляет метаданные клиента и сравнивает локальную версию политики с серверной. Если realtime-канал недоступен, сервер возвращает до нескольких ожидающих команд в поле `commands`.

```
POST /roy/api/v1/agent/heartbeat
Authorization: Bearer device-token
X-COPKA-ROY-Client-ID: 01a04411-2f4a-7b9e-9a31-2fb6b17e7810

{
  "hostname": "WIN-OPS-042",
  "platform": "Windows 11 Pro",
  "ip_address": "10.10.12.42",
  "agent_version": "1.0.0",
  "protocol_version": 1,
  "api_version": 1,
  "policy_uuid": "01a04422-9d3a-78f4-9e2c-990000000001",
  "policy_version": 17,
  "want_policy": false,
  "supports_interactive_monitoring": true
}
```

```
{
  "status": "ok",
  "client_id": "01a04411-2f4a-7b9e-9a31-2fb6b17e7810",
  "server_policy_version": 18,
  "policy_changed": true,
  "commands": [
    {
      "id": "01a04433-6f8d-7c62-9b13-2ed4bda60001",
      "command": "request_screenshot",
      "payload": {"reason": "manual"},
      "transport": "heartbeat"
    }
  ]
}
```

После обработки команды агент вызывает ACK endpoint. Статус может быть `delivered`, `running`, `completed`, `failed`, `unsupported`, `expired` или `cancelled`.

```
POST /roy/api/v1/agent/commands/01a04433-6f8d-7c62-9b13-2ed4bda60001/ack

{
  "status": "completed",
  "transport": "heartbeat",
  "result": {
    "message": "screenshot uploaded",
    "artifact_id": "01a04444-62d6-7560-9c2e-120000000001"
  }
}
```

Политики

Policy bundle возвращается по запросу `/policies/<client_id>` или внутри heartbeat, если клиент явно указал `want_policy` либо сервер обнаружил изменение версии. Сервер дополнительно проверяет, что client ID в пути совпадает с устройством из Bearer-токена.

```
GET /roy/api/v1/agent/policies/01a04411-2f4a-7b9e-9a31-2fb6b17e7810

{
  "schema_version": 1,
  "policy_version": 18,
  "policy_uuid": "01a04455-1405-730f-8a24-750000000001",
  "target_os_family": "windows",
  "policies": [
    {
      "id": "usb-control",
      "name": "USB контроль",
      "enabled": true,
      "priority": 100,
      "rules": []
    }
  ],
  "signature": "base64-ed25519-signature"
}
```

События агента

Пакетная загрузка событий используется для основной телеметрии. Сервер нормализует типы событий, назначает UUIDv7 при отсутствии `event_uuid`, применяет лимиты, создаёт инциденты по результату политики и игнорирует дубли при повторной отправке того же события.

Поле	Тип	Описание
`logs`	array	Список событий. Размер ограничен серверным лимитом batch.
`hostname`	string	Имя компьютера, обновляется в карточке клиента.
`platform`	string	ОС и платформа клиента.
`capabilities`	object	Фактические возможности установленного агента.
`policy_schema_versions`	array	Схемы политик, которые агент умеет применять.

```
POST /roy/api/v1/agent/logs

{
  "hostname": "WIN-OPS-042",
  "platform": "Windows 11 Pro",
  "agent_version": "1.0.0",
  "protocol_version": 1,
  "api_version": 1,
  "logs": [
    {
      "event_uuid": "01a04466-9871-706a-8a9d-8d0000000001",
      "source_table": "agent",
      "source_id": "clipboard:42",
      "event_type": "clipboard_event",
      "timestamp": "2026-08-27T08:12:00Z",
      "severity": "medium",
      "details": "clipboard text matched policy",
      "payload": {
        "channel": "clipboard",
        "policy_result": "deny",
        "matched_policy_name": "Персональные данные"
      }
    }
  ]
}
```

```
{
  "status": "ok",
  "received": 1,
  "inserted": 1,
  "duplicates": 0,
  "grouped": 0
}
```

Снимки экрана

Endpoint принимает multipart/form-data. Файл должен быть PNG или JPEG, сервер проверяет изображение, перекодирует его в PNG, сохраняет в объектное хранилище и применяет срок хранения рабочей области.

```
POST /roy/api/v1/agent/screenshot
Content-Type: multipart/form-data

file=@screen.png
timestamp=2026-08-27T08:13:00Z
```

Поиск файлов

Агент передаёт события поиска файлов в том же event model. Для простых результатов допускается массив `results`, который сервер преобразует в события `file_search_result`. Предпросмотр и загрузка файла выполняются через отдельные session endpoint, созданные веб-консолью.

```
POST /roy/api/v1/agent/file_search/results

{
  "events": [
    {
      "source_table": "file_search",
      "source_id": "run-42:path-hash",
      "event_type": "file_search_result",
      "timestamp": "2026-08-27T08:14:00Z",
      "severity": "low",
      "details": "matched *.xlsx",
      "payload": {
        "run_id": 42,
        "file_path": "C:/Users/user/Documents/report.xlsx",
        "size_bytes": 48120,
        "sha256": "..."
      }
    }
  ]
}
```

Обновление агента

Информация о текущем релизе доступна через `/version`. Если для клиента создано задание обновления, heartbeat возвращает update offer с grant ID, download key, целевой версией и сроком действия. Скачивание выполняется только по действующему grant, затем агент сообщает статус установки.

```
POST /roy/api/v1/agent/update/status

{
  "grant_id": "01a04477-412e-74da-93b5-4c0000000001",
  "status": "updated"
}
```

Remote shell events

Remote shell использует зашифрованный обмен. В API агента передаются только технические события с ciphertext и sequence; расшифровка выполняется на стороне браузера оператора и агента. Поддерживаемые события: `remote_shell_answer`, `remote_shell_output`, `remote_shell_status`.

```
POST /roy/api/v1/agent/remote_shell/events

{
  "event_name": "remote_shell_output",
  "payload": {
    "session_id": "01a04488-0189-71c8-8e77-260000000001",
    "algorithm": "ecdh-p256-hkdf-sha256-aes-256-gcm",
    "sequence": 7,
    "nonce": "base64-nonce",
    "ciphertext": "base64-ciphertext"
  }
}
```

Socket.IO

Socket.IO публикуется по ресурсу `/roy/socket.io`. Канал используется для быстрой доставки команд, live screen и интерактивных событий. Команды не считаются выполненными до ACK: если websocket недоступен или подтверждение не пришло, сервер оставляет команду в очереди и отдаёт её через heartbeat fallback.

Событие	Сторона	Применение
`join_client`	агент	Присоединение к комнате клиента после аутентификации.
`client_command`	сервер	Доставка команды агенту через realtime.
`command_ack`	агент	Подтверждение доставки или результата команды.
`live_screen_*`	агент/браузер	Служебный канал прямого экрана.
`remote_shell_*`	агент/браузер	Зашифрованный интерактивный канал remote shell.

Workspace API

Эти endpoint предназначены для пользователей веб-консоли и требуют обычной аутентифицированной сессии с доступом к рабочей области. Они не заменяют Agent API и не должны вызываться установленным агентом.

Метод	Маршрут	Назначение
GET	`/roy/api/v1/workspace`	Сведения о текущей рабочей области, версиях API и протокола.
GET	`/roy/api/v1/workspace/status`	Статус текущей рабочей области и ограничений.

Служебный API веб-консоли

Маршруты `/roy/internal/api/...` используются страницами веб-интерфейса. Это не внешний стабильный контракт для сторонних систем: формат может меняться вместе с UI, а доступ проверяется через пользовательскую сессию, роль и текущую рабочую область.

Группа	Маршруты	Применение
Клиент	`/client_status`, `/client_commands`, `/client_command`	Карточка клиента, отправка команд и просмотр результатов.
Скриншоты	`/client_screenshots`	Пагинация и обновление списка снимков клиента.
События	`/client_events/<category>`	Подгрузка событий клиента по категориям в UI.
Behavior metrics	`/behavior_metrics`	Метрики поведения для аналитических панелей.
File search	`/file_search/jobs`, `/runs`, `/found_files`, `/labels`, `/watchlist`	Настройка заданий поиска, запуск, результаты, метки, комментарии, watchlist и инциденты.
File transfer	`/preview_sessions`, `/download_sessions`	Создание сессий предпросмотра и контролируемой загрузки найденных файлов.
Remote shell	`/clients/<client_id>/remote_shell/...`	Старт, ввод, resize, stop и получение событий браузером оператора.

Коды ответов и ошибки

Код	Когда возвращается	Что делать клиенту
`200`	Запрос выполнен.	Обработать JSON или поток файла.
`400`	Некорректный JSON, неверный UUID, неподдерживаемый status или формат файла.	Не повторять без исправления данных.
`401`	Нет Bearer-токена, client ID не совпал с токеном или подпись недействительна.	Переинициализировать identity или запросить повторную регистрацию.
`403`	Ключ регистрации, grant обновления или права пользователя не подходят.	Остановить операцию и показать ошибку администратору.
`404`	Команда, пакет, клиент или сессия не найдены.	Синхронизировать состояние и не использовать устаревший идентификатор.
`409`	Повтор sequence/nonce, конфликт remote shell или неверная последовательность событий.	Не переиспользовать подпись; восстановить очередь событий.
`425`	Скачивание обновления разрешено позже.	Повторить после значения `Retry-After`.
`426`	Версия protocol/API не поддерживается.	Обновить агент или применить совместимый пакет.
`429`	Лимит запросов, событий или попыток скачивания.	Применить backoff и не наращивать частоту.
`503`	Недоступно хранилище, пакет агента или runtime.	Повторить после восстановления сервера.

Рекомендации по применению

- Агент должен считать registration key и device token секретами и не выводить их в обычные логи.
- Все рабочие запросы выполняются по HTTPS; прямой HTTP допустим только внутри доверенного reverse proxy контура.
- При сетевых сбоях агент повторяет отправку событий с тем же `event_uuid`, чтобы сервер мог убрать дубли.
- Heartbeat должен оставаться лёгким: крупные данные передаются через специализированные endpoint.