КриптоАРМ Подпись и шифрование

Внешний API

Интеграция сторонних приложений и веб-ресурсов с КриптоАРМ

ВЕРСИЯ ПРИЛОЖЕНИЯ

7.0.0

ДОКУМЕНТ АКТУАЛЕН НА

сентябрь 2026 г.

Содержание

API

1 Описание и возможности API

Внешний API КриптоАРМ: протокол cryptoarm://, доступные команды, сценарии вызова из веб-сервиса и локального приложения, формат JSON-RPC запросов и ответов.

API КриптоАРМ позволяет встроить подпись, шифрование и работу с сертификатами в веб-сервис или в локальное приложение. Приложение получает параметры вызова, открывает обычный экран операции, выполняет её после действия пользователя и возвращает результат вызывающей стороне.

Это руководство описывает протокол `cryptoarm://`, доступные команды, форматы запросов и ответов и сценарии интеграции.

Общая информация

Для взаимодействия используется зарегистрированный протокол `cryptoarm://`. Ссылку можно разместить на веб-странице, открыть из адресной строки браузера или передать из терминала.

Возможны две точки вызова:

- **сетевой вызов** — параметры операции приложение запрашивает у сервиса по HTTPS, результат отправляет ему же;
- **локальный вызов** — параметры приходят прямо в ссылке или в файле на этом же компьютере, результат сохраняется в указанный каталог.

API работает с сертификатами X.509. Операции OpenPGP через него недоступны — для их автоматизации используйте *командную строку* (Руководство пользователя, глава 20).

> 📌 **Примечание**: криптографические операции требуют действующей лицензии на КриптоАРМ. Временную лицензию на одну операцию можно передать в параметре `props.license`.

Доступные команды

Команда	Описание
`signAndEncrypt`	Подпись, архивирование, шифрование, проверка подписи, расшифрование, снятие подписи
`certificates`	Импорт и экспорт сертификатов, просмотр сведений о сертификате
`certrequests`	Создание запроса на сертификат (PKCS#10)
`diagnostics`	Сведения о рабочем месте: система, провайдеры, лицензии, личные сертификаты
`startView`	Открытие раздела приложения

Имя команды в ссылке нечувствительно к регистру.

Сценарий вызова из веб-приложения

1. **Инициация операции.** Пользователь выбирает на портале документы и действие — например, подпись.
2. **Формирование ссылки.** Портал отображает ссылку `cryptoarm://` с идентификатором транзакции или перенаправляет на неё браузер.
3. **Запуск приложения.** Система запускает КриптоАРМ, если он ещё не запущен, и передаёт ему ссылку.
4. **Проверка сервиса.** При первом обращении с этого адреса приложение показывает сертификат сервиса и запрашивает разрешение — см. Запрос от сетевого сервиса.
5. **Запрос параметров.** Приложение отправляет на указанный адрес JSON-RPC запрос `<команда>.parameters` и получает параметры операции.
6. **Подготовка операции.** Приложение загружает нужные файлы и открывает экран операции с переданными настройками.
7. **Подтверждение пользователем.** Пользователь проверяет, что именно запрошено, и запускает операцию.
8. **Отправка результата.** Результат уходит на адрес из `props.uploader` — в JSON или в `multipart/form-data`.

Сценарий вызова из локального приложения

1. **Инициация операции.** Пользователь выбирает файлы и действие в стороннем приложении.
2. **Формирование запроса.** Приложение готовит JSON-RPC объект с параметрами и открывает ссылку `cryptoarm://` со встроенным JSON либо со ссылкой на файл параметров.
3. **Запуск приложения.** Система запускает КриптоАРМ, если он ещё не запущен.
4. **Разрешение вызова.** Приложение показывает окно «Разрешить локальный API-вызов?».
5. **Подтверждение пользователем.** Пользователь проверяет параметры операции и запускает её.
6. **Сохранение результата.** Результат сохраняется в каталог, указанный в параметрах операции.

Формат ссылки для сетевого вызова

```
cryptoarm://<command>/<URL>?id=<id>
```

- `cryptoarm://` — зарегистрированный протокол;
- `<command>` — выполняемая команда;
- `<URL>` — адрес, по которому приложение запросит параметры операции и на который отправит результат;
- `id=<id>` — обязательный параметр. Идентификатор транзакции.

Пример сетевого вызова:

```
cryptoarm://signAndEncrypt/https://api.example.ru/cryptoarm/json?id=tx-42
```

Требования к адресу:

- только HTTPS: незащищённые соединения приложение отклоняет;
- ровно один непустой параметр `id`;
- имя пользователя и пароль в адресе не допускаются.

Формат ссылки для локального вызова

Вариант 1 — JSON в ссылке:

```
cryptoarm://<command>/<JSON>
```

Конверт приложение отличает от адреса по первому символу: значение, начинающееся с `{`, читается как JSON. Порядок ключей внутри конверта не важен. Объём встроенного JSON ограничен 2 МиБ. Состав `result` зависит от команды и описан на её странице.

```
cryptoarm://startView/{"jsonrpc":"2.0","id":"tx-42","result":{"uiView":"SIGN_AND_ENCRYPT"}}
```

Вариант 2 — файл параметров:

```
cryptoarm://<command>/file:///C:/work/request.json
```

Путь к файлу указывается в форме `file://`; сетевые узлы в таком адресе не поддерживаются. Входные файлы локального вызова также задаются через `file://`.

Дополнительные поля конверта в ссылке:

Ключ	Тип	Описание
`appName`	string	Необязательное поле. Название приложения, которое будет показано в окне разрешения.
`apiKey`	string	Необязательное поле. Проверочный код, который будет показан в окне разрешения.

Приложение показывает эти значения пользователю, но не проверяет их: решение принимает пользователь. Оба поля читаются только из конверта в ссылке: для варианта 2 разрешение запрашивается до чтения файла параметров, поэтому в окне показывается путь к файлу.

> ⚠️ **Важно**: сетевой вызов не может записать результат в локальный путь. Каталог результата задаётся только для локального вызова.

Кодирование ссылки

Хвост ссылки — адрес или JSON — передаётся либо как есть, либо целиком в процентной кодировке (`encodeURIComponent`). Приложение принимает оба вида:

```
cryptoarm://signAndEncrypt/https://api.example.ru/cryptoarm/json?id=tx-42
cryptoarm://signAndEncrypt/https%3A%2F%2Fapi.example.ru%2Fcryptoarm%2Fjson%3Fid%3Dtx-42
```

Откуда вызов	Как передавать
Ссылка на веб-странице	В процентной кодировке: браузер не примет кавычки и фигурные скобки
Командная строка, своё приложение	Как есть, взяв ссылку целиком в кавычки

В командной строке кавычки обязательны: без них пробелы и `&` разорвут ссылку на несколько аргументов, и приложение получит только её начало.

Аутентификация

Параметр `authType` и механизмы mTLS и OIDC не поддерживаются: ссылка с этим параметром отклоняется с ошибкой.

Подлинность сетевого сервиса подтверждается иначе — сертификатом сервиса, который пользователь видит и принимает при первом обращении, и последующей проверкой этого сертификата при каждом запросе. Подробности — в разделе Транспорт и доверие.

Токен доступа к файлам передаётся в `extra.token` команды `signAndEncrypt` и добавляется только к загрузке с адреса исходного вызова.

Описание запросов и ответов

Все запросы между приложением и сервисом соответствуют спецификации **JSON-RPC 2.0**. Транспорт — HTTP поверх TLS.

POST-запрос

Приложение выполняет POST-запросы с заголовками:

- `Content-Type: application/json; charset=utf-8`;
- `Content-Length` — длина тела запроса;
- `Accept: application/json`.

GET-запрос

GET-запросы используются для загрузки файлов — например, документов для подписи. Своих заголовков приложение к ним не добавляет; токен доступа передаётся параметром `accessToken` в адресе файла.

Перенаправления выключены: конечный адрес файла должен быть указан явно.

Запрос параметров

Ключ	Значение	Описание
`jsonrpc`	`2.0`	Версия протокола JSON-RPC. Всегда `2.0`.
`method`	`<команда>.parameters`	Имя метода — например, `signAndEncrypt.parameters`.
`id`	Идентификатор транзакции	Значение параметра `id` из ссылки.
`diagnostic`	Сведения о рабочем месте	Поля `VERSIONS`, `PROVIDERS` и `LICENSES` — см. `diagnostics`.

Для самой команды `diagnostics` поле `diagnostic` — пустой объект: сведения о рабочем месте не передаются до того, как пользователь разрешит операцию.

Ответ с параметрами

В теле ответа сервис возвращает конверт JSON-RPC. Заголовки ответа приложение не проверяет и разбирает тело как JSON:

Ключ	Значение	Описание
`jsonrpc`	`2.0`	Версия протокола JSON-RPC. Всегда `2.0`.
`result`	Параметры операции	Состав зависит от команды.
`id`	Идентификатор транзакции	Должен совпадать с `id` из ссылки.

Конверт проверяется строго:

- `jsonrpc` строго равен `"2.0"`;
- `id` совпадает с идентификатором из ссылки; `id: null` не принимается;
- присутствует ровно одно из полей: `result` или корректный `error`;
- пакетные (batch) запросы и скалярные значения вместо объекта не принимаются;
- неизвестные значения перечислимых полей и недопустимые сочетания операций отклоняются, а не заменяются значениями по умолчанию.

Объект Error

Если параметры выдать нельзя, сервис отвечает объектом ошибки:

Ключ	Тип	Описание
`code`	number	Код ошибки
`message`	string	Короткое описание ошибки
`data`	string/Object	Необязательное поле с дополнительными сведениями

Ответ на результат

Результат операции приложение отправляет как уведомление JSON-RPC. Сервис может ответить любым кодом `2xx`, включая `204 No Content`; тело ответа не требуется.

HTTP-коды

Код	Ошибка	Описание
200	OK	Успешный ответ и ответ с объектом ошибки
204	No Content	Для уведомлений без тела ответа
405	Method Not Allowed	Метод недоступен
415	Unsupported Media Type	`Content-Type` не `application/json`

Смотрите также

- Приложение и пользователь — окна согласия на стороне пользователя.
- Транспорт и доверие — HTTPS, доверенные сервисы, локальный вызов.
- Лимиты, статусы и ошибки — ограничения и завершение операции.
- *Выполнение операций в командной строке* (Руководство пользователя, глава 20) — другой способ автоматизации.

2 Приложение и пользователь

Окна согласия внешнего API КриптоАРМ: доступ сетевого сервиса, подтверждение операции, доверенные сервисы, фоновые разрешения и отзыв доверия.

Операция по ссылке `cryptoarm://` выполняется на рабочем месте пользователя и от его имени. Поэтому приложение открывает обычный экран операции, показывает, что именно запрошено, и выполняет операцию только после подтверждения.

Эта статья описывает интерфейс со стороны пользователя: что показано в окнах согласия и как отозвать доступ, который больше не нужен.

Запрос от сетевого сервиса

При первом обращении сервиса открывается окно **«Доступ к приложению КриптоАРМ»**.

В нём показаны:

Сведения	Что означают
Сервис	Адрес обратившегося ресурса
Сертификат ресурса	Владелец, издатель, серийный номер, срок действия, отпечаток SHA-256
Состав цепочки сертификации	Цепочка сертификата сервиса

Сертификат сервиса приложение дополнительно проверяет по своему хранилищу сертификатов и показывает результат двумя строками: **«Соединение защищено»** и **«Сертификат ресурса действителен»** либо **«Соединение не защищено»** и **«Сертификат ресурса недействителен»**. Обе строки отражают одну и ту же проверку сертификата: данные в любом случае передаются по HTTPS.

> ⚠️ **Важно**: разрешайте доступ только тем сервисам, которым доверяете. Сверьте адрес и отпечаток сертификата с тем, что сообщает сама организация. Если приложение показывает **«Сертификат ресурса недействителен»**, не разрешайте доступ.

Если сертификат ранее известного сервиса изменился, появится предупреждение **«Сертификат этого сервиса изменился. Проверьте новый отпечаток перед подтверждением доверия.»**

Нажмите **Сохранить**, чтобы разрешить доступ. Настройки ресурса сохранятся в профиле внешнего сервиса — см. Доверенные сервисы.

Запрос от локального приложения

Локальный вызов открывает окно **«Разрешить локальный API-вызов?»**. В нём показаны **«Приложение»** и **«Проверочный код»**, если вызывающая сторона передала их в конверте ссылки, а для вызова с файлом параметров — путь к нему в поле **«Файл параметров»**. Приложение не проверяет эти значения и не определяет, какая программа открыла ссылку: они помогают опознать вызов, но не подтверждают его источник.

Доступные решения:

- **Разрешить один раз** — только для текущего вызова;
- **Разрешить локальные вызовы до выхода** — до завершения работы приложения и для всех локальных вызовов, а не только для этой программы.

Разрешение локальных вызовов не сохраняется между запусками.

> ⚠️ **Важно**: разрешение «до выхода» не привязано к вызывающей программе. Пока приложение не закрыто, любой следующий локальный вызов — от другой программы или по ссылке, открытой в браузере, — выполняется без запроса. Выбирайте его только для серии вызовов подряд; в остальных случаях надёжнее **Разрешить один раз**.

Подтверждение операции

Перед выполнением открывается окно **«Подтвердите операцию»**. В нём показаны:

- **Команда** и **Действия** — что именно будет сделано;
- **Файлы** и их источник, для отсоединённой подписи — исходный файл;
- **Личный сертификат** — которым выполняется операция; если сервис его не задал, сертификат выбирает пользователь;
- **Параметры операции** — стандарт подписи, тип подписи, кодировки, алгоритм шифрования, подпись PAdES и видимый штамп, формат передачи результата;
- **Сетевые службы операции** — адреса службы штампов времени и службы проверки статуса сертификата, к которым обратится приложение;
- **Назначение результата** — куда будет передан результат.

Нажмите **Выполнить операцию**, чтобы запустить её, или **Отмена**, чтобы отказаться.

Параметры, которые сервис задал явно, показаны без возможности изменения; остальные пользователь может изменить перед запуском.

> 💡 **Совет**: перед подписанием откройте файлы и убедитесь, что на подпись отправлены нужные документы.

Разрешение на диагностику

Запрос сведений о рабочем месте открывает окно **«Диагностика рабочего места»**. В блоке **«Запрашиваемые сведения»** перечислено, что именно запросил сервис: сведения о системе, версии КриптоАРМ, КриптоПро CSP и OpenSSL, доступность провайдеров, сведения о лицензиях, сведения о личных сертификатах.

Нажмите **Разрешить** или **Отклонить**. Для сетевого сервиса в этом окне можно включить **«Разрешить проведение диагностики приложения в фоновом режиме»** — тогда последующие запросы диагностики от этого сервиса будут выполняться без подтверждения.

Доверенные сервисы

Сервисы, которым вы разрешили доступ, сохраняются как профили внешних сервисов. Их список находится в разделе **«Подпись и шифрование»** в панели профилей — под списком профилей подписи, в группе **«Доверенные сервисы»**.

Щелчок по записи открывает вкладку **«Профиль внешнего сервиса»** с двумя вкладками:

Вкладка	Содержание
Настройки профиля	Параметры подписи и шифрования, с которыми выполняются операции этого сервиса
Параметры ресурса	Адрес ресурса, статус соединения, сертификат ресурса и его цепочка, фоновые разрешения

Для каждого сервиса создаётся отдельный профиль подписи. Операции этого сервиса выполняются по нему, а не по профилю, с которым работал пользователь в последний раз.

Фоновые разрешения

Два действия сервис может выполнять без подтверждения, если это разрешено явно:

Разрешение	Что позволяет
Фоновая диагностика	Получать сведения о рабочем месте: провайдеры, версии, лицензии
Фоновая проверка подписи	Проверять подписи без запроса подтверждения

Оба разрешения выключены по умолчанию. Включить их можно в окне согласия в блоке **«Дополнительно»** или позже — на вкладке **«Параметры ресурса»** профиля внешнего сервиса.

> ⚠️ **Важно**: фоновая диагностика действует, пока вы не выключите её на вкладке **«Параметры ресурса»** или не удалите профиль сервиса. Пока она включена, сервис без запроса получает при каждом обращении те сведения, которые запросит, — вплоть до состава рабочего места и данных обо всех личных сертификатах: владельца, организации, криптопровайдера, сроков действия и наличия закрытого ключа. Полный перечень — в разделе Сведения о личных сертификатах.

Подпись, шифрование, расшифрование, снятие подписи, импорт и экспорт сертификатов и создание запроса всегда требуют подтверждения.

Отзыв доверия

1. Откройте панель профилей в разделе **«Подпись и шифрование»**.
2. В группе **«Доверенные сервисы»** найдите нужный сервис.
3. Удалите запись и подтвердите действие в окне **«Удаление профиля внешнего сервиса»**.

Появится сообщение **«Профиль внешнего сервиса „...“ удалён»**. Вместе с профилем удаляются сохранённый сертификат сервиса и его настройки: при следующем обращении сервис снова должен будет получить разрешение.

Смотрите также

- Описание и возможности API — команды и форматы вызова.
- Транспорт и доверие — как проверяется сертификат сервиса.
- *Профили подписи* (Руководство пользователя, глава 14) — где хранятся профили.
- *Общие настройки* (Руководство пользователя, глава 10) — остальные настройки приложения.

3 Транспорт и доверие

Как внешний API КриптоАРМ проверяет сервис: HTTPS, закрепление сертификата, профиль доверенного сервиса, адреса файлов и токен, локальный вызов и права на фоновые операции.

Приложение выступает HTTP-клиентом: оно само обращается к сервису за параметрами и само отправляет результат. Локального сервера, принимающего входящие соединения, у него нет.

Сетевой вызов

Формат ссылки:

```
cryptoarm://<command>/<https-url>
```

Адрес должен содержать ровно один непустой параметр `id` — идентификатор всей транзакции. Он используется в запросе параметров, в ответе сервиса и в результате операции.

```
cryptoarm://diagnostics/https://api.example.ru/cryptoarm/json?id=tx-42
```

После подтверждения доверия приложение отправляет запрос параметров:

```
{
  "jsonrpc": "2.0",
  "method": "diagnostics.parameters",
  "id": "tx-42",
  "diagnostic": {}
}
```

Для остальных команд поле `diagnostic` содержит `VERSIONS`, `PROVIDERS` и `LICENSES`.

Сервис отвечает параметрами команды:

```
{
  "jsonrpc": "2.0",
  "id": "tx-42",
  "result": {
    "operation": ["VERSIONS", "LICENSES"]
  }
}
```

Требования к ответу перечислены в разделе Описание запросов и ответов. Оформление видимого штампа PAdES проверяется менее строго, чем остальные поля: неизвестные поля игнорируются, а непригодные декоративные значения — цвет, логотип, масштаб — заменяются оформлением по умолчанию.

Результат отправляется как уведомление JSON-RPC. Получатель может вернуть любой код `2xx`, включая `204 No Content`; тело ответа не требуется.

Доверенный сервис

Сетевой вызов работает только по HTTPS; имя пользователя и пароль в адресе запрещены. При первом обращении с нового адреса приложение показывает сертификат сервиса и предлагает отменить вызов или сохранить сервис как доверенный.

Доверие определяется тремя значениями:

- точный адрес ресурса — схема, узел и порт;
- отпечаток SHA-256 открытого ключа сертификата (SPKI pin);
- отпечаток самого сертификата.

Обычная проверка цепочки TLS и имени узла сохраняется, а закреплённый ключ дополнительно проверяется при каждом запросе к сервису. Смена сертификата требует нового решения пользователя.

В окне согласия сертификат дополнительно проверяется по хранилищу сертификатов приложения: цепочка, срок действия, пригодность сертификата для TLS-сервера и соответствие имени узла. Результат этой проверки показывается пользователю вместе со сведениями о сертификате и цепочке.

Сохранённый сервис отображается в группе **«Доверенные сервисы»** панели профилей подписи — см. Доверенные сервисы. После удаления профиля доверие, закреплённый ключ и связанный профиль удаляются, и следующее обращение снова требует подтверждения.

Профиль сервиса и настройки операции

Вместе с доверием создаётся отдельный профиль подписи для этого сервиса. Итоговые настройки операции вычисляются в порядке:

1. Значения по умолчанию, заданные приложением.
2. Профиль сервиса.
3. Явные параметры вызова.
4. Изменения пользователя в незаблокированных полях.

Профиль, с которым пользователь работал в последний раз вручную, не используется. Из профиля сервиса не переносятся состав операций, локальная выходная папка, удаление исходных файлов, МЧД и назначение результата: они задаются транзакцией API.

Адреса файлов и токен

Адреса входных файлов и адрес получателя результата проверяются отдельно от адреса вызова. Перенаправления выключены: конечный адрес должен быть указан явно, а новый адрес требует собственного подтверждения доверия.

Значение `extra.token` добавляется как параметр `accessToken` только к адресу входного файла с того же ресурса, что и сам вызов. На другой ресурс и на получателя результата токен не переносится. Для файлов на другом ресурсе используйте предподписанные ссылки либо передайте нужный параметр доступа прямо в адресе файла.

Службы штампов времени (TSP) и проверки статуса сертификата (OCSP) — исключение: к ним обращается не приложение, а криптографический провайдер. Принимается любой корректный адрес HTTP или HTTPS, включая внутренние адреса и адреса с учётными данными; `ocspURL` требует `tspURL`. Оба адреса показываются пользователю в окне подтверждения операции.

Локальный вызов

Вместо адреса HTTPS ссылка может содержать JSON-конверт или адрес `file://` файла с параметрами:

```
cryptoarm://startView/{"jsonrpc":"2.0","id":"tx-42","result":{"uiView":"SIGN_AND_ENCRYPT"}}
cryptoarm://signAndEncrypt/file:///C:/work/request.json
```

Конверт имеет вид `{ "jsonrpc": "2.0", "id": ..., "result": ... }`. Входные файлы задаются адресами `file://`. Ссылку можно передать и в процентной кодировке — правило описано в разделе Кодирование ссылки.

Каталог результата задаётся обычным путём или адресом `file://`:

Команда	Где указывается
`signAndEncrypt`	`props.localResultParams.savePath`
`certificates`	`localResultParams` рядом с `operation` и `props`
`certrequests` и `diagnostics`	Рядом с `operation` и `props` либо внутри `props`

Для локального вызова `signAndEncrypt` в `props.uploader` принимается как адрес `file://`, так и адрес HTTPS. Значение `saveResultsSeparately: true` создаёт отдельный файл результата для каждого идентификатора файла. Каталог может ещё не существовать — приложение создаст его после подтверждения вызова.

> ⚠️ **Важно**: сетевой вызов не может записать результат в локальный путь или в `file://`. Разрешение **«Разрешить локальные вызовы до выхода»** действует только до завершения работы приложения и не сохраняется.

Подтверждение пользователем

Доверие к сервису не даёт права выполнять криптографические операции без участия человека. Подпись, шифрование, расшифрование, снятие подписи, импорт и экспорт сертификатов и создание запроса открывают соответствующий экран или диалог приложения.

Фоновая диагностика и фоновая проверка подписи разрешаются отдельными настройками доверенного сервиса, выключенными по умолчанию. Фоновая проверка действует только для одиночной операции `VERIFYSIGN`.

ПИН-код и временная лицензия из `props.license` хранятся только в памяти одной операции: в журнал и в лог они не попадают.

Смотрите также

- Описание и возможности API — команды и форматы вызова.
- Приложение и пользователь — окна согласия и доверенные сервисы.
- Лимиты, статусы и ошибки — ограничения транспорта.

4 Команда signAndEncrypt

Команда signAndEncrypt внешнего API КриптоАРМ: подпись, архивирование, шифрование, проверка подписи, расшифрование и снятие подписи. Форматы запросов, параметры и результаты.

Команда запускает подпись, архивирование и шифрование документов либо одну обратную операцию — проверку подписи, расшифрование или снятие подписи. Перед запуском пользователь видит обычный экран операции с файлами, сертификатами и настройками.

Общая информация

Команда работает с сертификатами X.509. Операции OpenPGP через внешний API недоступны.

> 📌 **Примечание**: выполнение операции требует действующей лицензии на КриптоАРМ. Временную лицензию на одну операцию можно передать в `props.license`.

Формат ссылки

```
cryptoarm://signAndEncrypt/<URL>?id=<id>
```

Пример сетевого вызова:

```
cryptoarm://signAndEncrypt/https://api.example.ru/cryptoarm/json?id=tx-42
```

Формат локального вызова описан в разделе Формат ссылки для локального вызова.

Операции

Поле `operation` — непустой массив без повторов:

Прямые операции (можно комбинировать):

Значение	Описание
`SIGN`	Подпись
`ARCHIVE`	Архивирование
`ENCRYPT`	Шифрование

Обратные операции (выполняются по одной):

Значение	Описание
`UNSIGN`	Снятие подписи
`DECRYPT`	Расшифрование

Проверка подписи:

Значение	Описание
`VERIFYSIGN`	Проверка подписи

Ограничения:

- прямые и обратные операции не смешиваются;
- `VERIFYSIGN` и `UNSIGN` выполняются только по одной;
- отдельной операции `UNZIP` нет: ZIP-слой снимается вместе с проверкой или расшифрованием.

Получение параметров операции

Формат запроса

Получив команду, приложение отправляет на указанный адрес запрос параметров.

Ключ	Значение	Описание
`jsonrpc`	`2.0`	Версия протокола JSON-RPC. Всегда `2.0`.
`method`	`signAndEncrypt.parameters`	Имя метода. Всегда `signAndEncrypt.parameters`.
`id`	Идентификатор транзакции	Значение параметра `id` из ссылки.
`diagnostic`	Сведения о рабочем месте	Поля `VERSIONS`, `PROVIDERS` и `LICENSES`.

```
{
  "jsonrpc": "2.0",
  "method": "signAndEncrypt.parameters",
  "id": "tx-42",
  "diagnostic": {
    "VERSIONS": { "csp": "5.0.13000", "cryptoarm": "1.2.0", "openssl": "3.0.13" },
    "PROVIDERS": { "GOST2012_256": true, "GOST2012_512": true, "openssl": true },
    "LICENSES": {
      "csp": { "status": true, "type": "client", "expiration": "" },
      "cryptoarm": { "status": true, "type": "Permanent", "expiration": "" }
    }
  }
}
```

Формат ответа

Пример для прямых операций:

```
{
  "jsonrpc": "2.0",
  "id": "tx-42",
  "result": {
    "operation": ["SIGN", "ENCRYPT"],
    "props": {
      "headerText": "Подписание документов",
      "descriptionText": "Пакет документов на согласование",
      "files": [
        {
          "id": "document-1",
          "name": "document.pdf",
          "url": "https://api.example.ru/files/document.pdf"
        }
      ],
      "uploader": "https://api.example.ru/cryptoarm/result",
      "uploadMethod": "json",
      "extra": {
        "signType": 0,
        "signStandard": "cades-bes",
        "encryptAlgorithm": 0,
        "returnFiles": true
      }
    }
  }
}
```

Пример для проверки подписи:

```
{
  "jsonrpc": "2.0",
  "id": "tx-42",
  "result": {
    "operation": ["VERIFYSIGN"],
    "props": {
      "files": [
        {
          "id": "signature-1",
          "name": "document.pdf.sig",
          "url": "https://api.example.ru/files/document.pdf.sig",
          "urlDetached": "https://api.example.ru/files/document.pdf"
        }
      ],
      "uploader": "https://api.example.ru/cryptoarm/result",
      "extra": { "printReport": true }
    }
  }
}
```

Файлы

Файл описывается объектом:

Поле	Тип	Описание
`id`	string	Идентификатор файла. Возвращается в результате и используется в multipart.
`name`	string	Имя файла.
`url`	string	Адрес файла: HTTPS для сетевого вызова, `file://` — для локального.
`urlDetached`	string	Необязательное поле. Исходный документ для отсоединённой подписи.

Поля `props.files` и `props.archive` взаимоисключающие, и одно из них обязательно. В `archive` передаётся ZIP с входными файлами: он распаковывается до запуска операции. Идентификаторы файлов должны быть уникальны.

Параметры операции

Поле	Описание
`props.headerText`	Заголовок операции, не более 40 символов
`props.descriptionText`	Описание операции, не более 120 символов
`props.files`	Массив входных файлов
`props.archive`	ZIP с входными файлами вместо `props.files`
`props.license`	Временная лицензия на одну операцию
`props.uploader`	Адрес, на который отправляется результат; обязателен для сетевого вызова
`props.uploadMethod`	`json` (по умолчанию) или `multipart/form-data`
`props.localResultParams.savePath`	Каталог результата; только для локального вызова
`props.localResultParams.saveResultsSeparately`	Отдельный файл результата для каждого входного файла
`props.extra`	Настройки операции — см. ниже

Параметры extra

Поле	Значения
`signType`	`0` — присоединённая подпись (по умолчанию), `1` — отсоединённая
`signStandard`	`0`, `cms` или `cades-bes` (по умолчанию), `1` или `cades-xlt1`, `2` или `cades-t`, `3` или `cades-a`
`signEncoding`	`0`, `PEM` или `BASE-64`; `1` или `DER`
`encryptEncoding`	`0`, `PEM` или `BASE-64`; `1` или `DER`
`encryptAlgorithm`	`0` — ГОСТ 28147-89 (по умолчанию), `1` — Магма, `2` — Кузнечик
`signatureExtension`	`sig` (по умолчанию), `p7s`, `sgn`, `sign`, `bin`, `pem`

Поле	Значения
`encryptionExtension`	`enc` (по умолчанию), `p7m`, `p7e`, `pem`
`signerChainIncludeKind`	`personal` — только сертификат подписанта (по умолчанию), `full` — вся цепочка
`timestampOnSign`	`true` вместе с CAdES-BES включает CAdES-T; `false` несовместимо с CAdES-T, XLT1 и A
`tspURL`, `ocspURL`	Адреса служб штампа времени и проверки статуса сертификата; `ocspURL` требует `tspURL`
`isSignaturePades`	Подпись PDF в формате PAdES
`isPdfCertificationSign`	Сертифицирующая подпись PDF
`isAddStampToPdf`	Отдельная печатная копия PDF со штампом подписи
`printReport`	PDF-отчёт о проверке подписи
`returnFiles`	`true` (по умолчанию) — возвращать содержимое файлов результата
`showSettingsBeforeOperation`	`true` (по умолчанию) открывает настройки операции, `false` открывает подтверждение запуска без экрана настроек
`signCertificate`	Сертификат подписанта
`encryptCertificates`	Сертификаты получателей
`MCHD`	Машиночитаемая доверенность
`token`	Токен доступа к входным файлам с адреса исходного вызова
`pin`	ПИН-код контейнера закрытого ключа. Хранится только в памяти операции

Числовые значения принимаются и строками: `"1"` равнозначно `1`.

Неизвестное значение перечислимого поля — ошибка: приложение не заменяет его значением по умолчанию.

Сертификаты

Сертификат подписанта задаётся одним из трёх способов:

Селектор	Пример
Отпечаток SHA-1	`{ "hash": "b0a1…" }`
Сам сертификат	`{ "x509": "<DER Base64>" }`
Издатель и серийный номер	`{ "issuerName": "CN=Example CA", "serial": "01AB" }`

```
{
  "extra": {
    "signCertificate": { "hash": "b0a1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9" }
  }
}
```

Получатели шифрования передаются в `encryptCertificates` — массивом сертификатов в формате DER, закодированном в Base64.

Все явно заданные сертификаты должны однозначно определяться в локальном хранилище; частично найденный список не используется. Если сертификат не задан, приложение предлагает выбрать его в обычной форме операции. Переданные сервисом сертификаты показываются без возможности изменения.

Подпись PDF и штамп

Оформление видимого штампа задаётся объектом `PAdESStampAppearance` — для подписи PAdES — и объектом `signStampAppearance` — для отдельной печатной копии со штампом.

Блок `mockupSettings` — вид штампа:

Поле	Значения
`elementsColor`	Цвет текста и рамки в формате `#rrggbb`
`backgroundColor`	Цвет фона
`addBackground`	Заливать фон. По умолчанию `true`
`addBorders`	Рисовать рамку. По умолчанию `true`
`pixelRatio`	Масштаб отрисовки от 0.5 до 4. По умолчанию 2
`logotype`	Логотип: `…` или `…`
`isHiddenLogo`	Скрыть логотип

Блок `requisitesSettings` — состав реквизитов:

Поле	Значения
`centralDisplayType`	`arbitrary` — произвольный состав центрального блока; иначе используется состав по ГОСТ
`central`	Реквизиты центрального блока
`left`, `right`	Реквизиты боковых блоков
`position`	`left`, `right` или `leftAndRight`

Элемент списка реквизитов: `{ "dataKey": "…", "isEnabled": true, "editedValue": "…", "title": "…" }`. Отключённые и повторяющиеся реквизиты отбрасываются. Доступные значения `dataKey`: `serialNumber`, `validityTimeSpan`, `subjectFriendlyName`, `individual`, `jobPosition`, `issuerFriendlyName`,

`organizationName`, `mark`, `description`, `location`, `signingTime`, `MCHDNumber`, `hash`, `signatureAlgorithm`.

Размещение штампа для PAdES задаётся объектом `signStampPAdES`:

Поле	Значения
`pagesSelection`	`all`, `last` или `some`
`pageNumbers`	Массив номеров страниц; обязателен при `some`
`isDisplayedOnAllPages`	Показывать штамп на всех страницах
`pdfMarkedArea.stampSize`	`width` и `height` штампа в пунктах PDF
`pdfMarkedArea.dimensionsDocument`	`width`, `height`, `horizontalPadding`, `verticalPadding`
`pdfMarkedArea.placementOnPage`	`horizontal`: `left`, `center`, `right`; `vertical`: `top`, `center`, `bottom`
`pdfMarkedArea.orientationDegrees`	Поворот: `0`, `90`, `180` или `270`

Для печатной копии страницы задаются внутри `signStampAppearance`: `pageSelection` и строка `pageNumbers` вида `"1,3-5"`.

Устаревший способ размещения `signStampPosition` с полями `width`, `height`, `horizontalPadding` и `verticalPadding` также принимается; его значения задаются в миллиметрах, тогда как размеры в `signStampPAdES` — в пунктах PDF. По умолчанию штамп имеет размер 102,3 × 39,16 мм и отступы 66,6 мм по горизонтали и 20 мм по вертикали.

Недопустимые размеры, выбор страниц, поворот и размещение возвращают ошибку и не заменяются значениями по умолчанию. Значения по умолчанию подставляются только для декоративных полей — цвета, логотипа и масштаба.

Машиночитаемая доверенность

МЧД передаётся в `extra.MCHD` вместе с отсоединёнными подписями:

```
{
  "MCHD": {
    "xml": {
      "id": "mchd",
      "name": "power.xml",
      "url": "https://api.example.ru/files/power.xml"
    },
    "signatures": [
      {
        "id": "mchd-sign",
        "name": "power.xml.sig",
        "url": "https://api.example.ru/files/power.xml.sig"
      }
    ]
  }
}
```

Требуется файл XML и хотя бы одна подпись; идентификаторы всех файлов МЧД должны быть уникальны.

Настройки и подтверждение

Явно переданные через API тип и стандарт подписи, кодировки, расширения файлов, алгоритм шифрования, параметры PAdES и МЧД и сертификаты заблокированы для изменения. Если параметр не передан, начальное значение берётся из профиля сервиса, и пользователь может его изменить.

Состав операций, входные файлы и назначение результата задаются транзакцией API и не редактируются. Значение `showSettingsBeforeOperation: false` пропускает экран настроек, но не подтверждение запуска.

Отправка результатов

Прямые и обратные операции

Результат отправляется методом `signAndEncrypt.outDirectResults`.

Ключ	Значение	Описание
`jsonrpc`	`2.0`	Версия протокола JSON-RPC. Всегда `2.0`.
`method`	`signAndEncrypt.outDirectResults`	Имя метода
`params`	Объект результата	Идентификатор транзакции, статус и список файлов

```
{
  "jsonrpc": "2.0",
  "method": "signAndEncrypt.outDirectResults",
  "params": {
    "id": "tx-42",
    "status": "Completed",
    "directResults": [
      {
        "id": "document-1",
        "out": "<Base64>"
      }
    ]
  }
}
```

Проверка подписи

Результат проверки отправляется методом `signAndEncrypt.verifySignResults` в поле `verifySignResults`. Содержимое файлов при проверке не возвращается: вместо него передаются сведения о подписи, а при `printReport: true` — PDF-отчёт в поле `outReport`.

```
{
  "jsonrpc": "2.0",
  "method": "signAndEncrypt.verifySignResults",
  "params": {
    "id": "tx-42",
    "status": "Completed",
    "verifySignResults": [
      {
        "id": "signature-1",
        "signValid": false,
        "isSignaturePades": false,
        "signers": [
          {
            "isValid": false,
            "isDetached": true,
            "isCades": true,
            "cadesType": 1,
            "signingTime": 1594209848000,
            "signatureAlgorithm": "1.2.643.7.1.1.3.2",
            "signatureDigestAlgorithm": "1.2.643.7.1.1.2.2",
            "signerCertificate": {
              "hash": "b0a1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9",
              "issuerFriendlyName": "Example CA",
              "issuerName": "CN=Example CA",
              "subjectFriendlyName": "Example User",
              "subjectName": "CN=Example User",
              "status": true,
              "serial": "01AB",
              "notBefore": 1593603200000,
              "notAfter": 1725139200000
            }
          }
        ]
      }
    ]
  }
}
```


> 💡 **Примечание**: недействительная подпись — это результат проверки, а не ошибка. Конверт имеет `status: "Completed"`, а `signValid` и `signers[].isValid` — значение `false`.

Передача файлов

При `uploadMethod: "json"` и `returnFiles: true` содержимое файлов передаётся в поле `out` в Base64. Base64 увеличивает объём передачи примерно на треть, поэтому для больших результатов используйте `multipart/form-data` или `returnFiles: false`.

При `uploadMethod: "multipart/form-data"` конверт JSON передаётся в поле `operation-response`, а файлы — отдельными частями, имя каждой части равно идентификатору исходного файла. Отчёт о проверке и печатная копия со штампом остаются полями конверта в Base64.

Значение `returnFiles: false` оставляет метаданные результата без содержимого файлов.

Сохранение результата локально

Для локального вызова результат сохраняется в каталог из `props.localResultParams.savePath` либо по адресу `file://` в `props.uploader`. Имена файлов:

Операция	Имя файла
Прямая и обратная	`direct-result-<id>.json`
Проверка подписи	`verify-result-<id>.json`

При `saveResultsSeparately: true` для каждого входного файла создаётся отдельный файл `direct-result-<id>-<fileId>.json`. Существующие файлы не перезаписываются: к имени добавляется порядковый номер.

Справочник полей результата

Объект результата файла

Поле	Тип	Описание
`id`	string	Идентификатор входного файла
`out`	string	Содержимое результата в Base64
`outReport`	string	PDF-отчёт о проверке подписи в Base64
`outSignStamp`	string	Печатная копия PDF со штампом в Base64
`signValid`	boolean	Все подписи файла действительны
`timestampValid`	boolean	Штамп времени действителен
`isSignaturePades`	boolean	Подпись встроена в PDF
`signers`	Object[]	Сведения о подписантах
`error`	string	Код ошибки для этого файла

Статус подписанта

Поле	Тип	Описание
`isValid`	boolean	Подпись действительна
`signingTime`	number	Время подписания, Unix-время в миллисекундах
`verifyingTime`	number	Время проверки, Unix-время в миллисекундах
`isDetached`	boolean	Отсоединённая подпись
`isCades`	boolean	Подпись в формате CAdES
`cadesType`	number	Тип CAdES
`isSignaturePades`	boolean	Подпись встроена в PDF

Поле	Тип	Описание
`signatureAlgorithm`	string	Идентификатор алгоритма подписи
`signatureDigestAlgorithm`	string	Идентификатор алгоритма хеширования
`signerCertificate`	Object	Сертификат подписанта

Сертификат подписанта

Поле	Тип	Описание
`hash`	string	Отпечаток сертификата SHA-1
`issuerName`	string	Издатель
`issuerFriendlyName`	string	Издатель в удобочитаемом виде
`subjectName`	string	Владелец
`subjectFriendlyName`	string	Владелец в удобочитаемом виде
`serial`	string	Серийный номер
`status`	boolean	Результат проверки сертификата
`provider`	string	Криптопровайдер
`notBefore`	number	Начало срока действия, Unix-время в миллисекундах
`notAfter`	number	Окончание срока действия, Unix-время в миллисекундах
`rootCAMinComSvyaz`	boolean	Цепочка завершается корневым сертификатом Минцифры

Смотрите также

- Описание и возможности API — общий формат вызова.
- Транспорт и доверие — адреса файлов, токен, локальный вызов.
- Лимиты, статусы и ошибки — ограничения и завершение операции.
- *Подпись и шифрование* (Руководство пользователя, глава 15) — те же операции в интерфейсе.

5 Команда certificates

Команда certificates внешнего API КриптоАРМ: импорт сертификата в локальное хранилище, экспорт выбранных сертификатов и просмотр сведений. Форматы запросов и ответов.

Команда импортирует сертификат X.509 в локальное хранилище, экспортирует выбранные сертификаты либо показывает сведения о переданном сертификате.

Общая информация

Все три операции требуют действия пользователя: приложение открывает соответствующий экран, и без подтверждения хранилище не изменяется и сертификаты не выдаются.

Закрытые ключи через эту команду не передаются.

Формат ссылки

```
cryptoarm://certificates/<URL>?id=<id>
```

Формат локального вызова описан в разделе Формат ссылки для локального вызова.

Параметры

Параметры содержат поле `operation` и объект `props`:

Поле	Значение
`operation`	`import`, `export` или `information`
`props.headerText`	Необязательный заголовок, не более 40 символов
`props.descriptionText`	Необязательное описание, не более 120 символов
`props.store`	Массив хранилищ
`props.multy`	Разрешить выбор нескольких сертификатов при экспорте; по умолчанию `false`
`props.certificateBase64`	Сертификат в формате DER, закодированном в Base64 — для `import` и `information`
`localResultParams.savePath`	Каталог результата; только для локального вызова

Хранилища:

Значение	Раздел приложения
`MY`	Личные сертификаты
`AddressBook`	Сертификаты других пользователей
`CA`	Промежуточные удостоверяющие центры
`ROOT`	Корневые удостоверяющие центры
`CA_OR_ROOT`	Удостоверяющие центры — без разделения на промежуточные и корневые

Повторяющиеся значения `store` отбрасываются. Неизвестное хранилище — ошибка: список не расширяется до всех сертификатов.

Импорт сертификата

```
{
  "jsonrpc": "2.0",
  "id": "tx-import",
  "result": {
    "operation": "import",
    "props": {
      "store": ["MY"],
      "certificateBase64": "<DER Base64>"
    }
  }
}
```

Приложение показывает сертификат и хранилища, допустимые для него: пользовательский сертификат — **Личные** и **Другие пользователи**, самоподписанный сертификат УЦ — **Корневые удостоверяющие центры**, остальные сертификаты УЦ — **Промежуточные удостоверяющие центры**; для сертификата УЦ доступно также хранилище **Удостоверяющие центры**. Значение `props.store[0]` подставляется как предварительный выбор, но окончательное хранилище выбирает пользователь.

Результат:

```
{
  "jsonrpc": "2.0",
  "method": "certificates.import",
  "params": {
    "id": "tx-import",
    "status": "Completed"
  }
}
```

Экспорт сертификатов

```
{
  "jsonrpc": "2.0",
  "id": "tx-export",
  "result": {
    "operation": "export",
    "props": {
      "store": ["MY", "AddressBook"],
      "multy": true
    }
  }
}
```

Если `props.store` не задан, предлагаются личные сертификаты. Пользователь выбирает один или несколько сертификатов, и приложение отправляет их в формате DER, закодированном в Base64, вместе со сведениями о владельце:

```
{
  "jsonrpc": "2.0",
  "method": "certificates.base64",
  "params": {
    "id": "tx-export",
    "certificates": [
      {
        "certificateBase64": "<DER Base64>",
        "hash": "b0a1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9",
        "issuerFriendlyName": "Example CA",
        "issuerName": "CN=Example CA",
        "notAfter": "2027-07-13T10:00:00.000Z",
        "notBefore": "2026-07-13T10:00:00.000Z",
        "rootCAMinComSvyaz": false,
        "subjectFriendlyName": "Example User",
        "subjectName": "CN=Example User",
        "status": true,
        "serial": "01AB"
      }
    ]
  }
}
```

Просмотр сведений о сертификате

```
{
  "jsonrpc": "2.0",
  "id": "tx-information",
  "result": {
    "operation": "information",
    "props": {
      "certificateBase64": "<DER Base64>"
    }
  }
}
```

Приложение открывает карточку сертификата в отдельной вкладке и отправляет:

```
{
  "jsonrpc": "2.0",
  "method": "certificates.information",
  "params": {
    "id": "tx-information",
    "status": "Completed"
  }
}
```

Для сетевого вызова результат уходит на адрес исходного вызова; локальный вызов сохраняет тот же конверт в файл `certificates-<id>.json` в каталоге `localResultParams.savePath`.

Справочник полей результата

Поле	Тип	Описание
`certificateBase64`	string	Сертификат в формате DER, закодированном в Base64
`hash`	string	Отпечаток сертификата SHA-1
`issuerName`	string	Издатель
`issuerFriendlyName`	string	Издатель в удобочитаемом виде
`subjectName`	string	Владелец
`subjectFriendlyName`	string	Владелец в удобочитаемом виде
`serial`	string	Серийный номер
`notBefore`	string	Начало срока действия
`notAfter`	string	Окончание срока действия
`status`	boolean	Результат проверки цепочки сертификата
`rootCAMinComSvyaz`	boolean	Цепочка завершается корневым сертификатом Минцифры

Смотрите также

- Описание и возможности API — общий формат вызова.
- Команда certrequests — создание запроса на сертификат.
- *Управление сертификатами* (Руководство пользователя, глава 21) — те же операции в интерфейсе.

6 Команда certrequests

Команда certrequests внешнего API КриптоАРМ: создание запроса PKCS#10 по шаблону JSONTemplate или по существующему сертификату. Поля шаблона, ограничения и формат результата.

Команда создаёт запрос на сертификат в формате PKCS#10. Сервис передаёт шаблон запроса, а алгоритм, контейнер и доступные поля пользователь проверяет в обычной форме создания запроса.

Общая информация

Поддерживается единственная операция — `GENERATE`. Запрос всегда создаётся в кодировке DER с расширением `.csr`. Создание самоподписанного сертификата и выгрузка данных владельца в XML через эту команду недоступны.

Ограничения шаблона проверяются повторно после действий пользователя, поэтому заблокированное или обязательное поле нельзя обойти изменением формы.

Формат ссылки

```
cryptoarm://certrequests/<URL>?id=<id>
```

Формат локального вызова описан в разделе Формат ссылки для локального вызова.

Параметры

Поле	Значение
`operation`	Только `GENERATE`
`props.headerText`	Необязательный заголовок, не более 40 символов
`props.descriptionText`	Необязательное описание, не более 120 символов
`props.templateType`	`JSONTemplate` или `CertificateTemplate`
`props.template`	Объект шаблона выбранного типа
`localResultParams.savePath`	Каталог результата; допустим также внутри `props`

Шаблон JSONTemplate

```
{
  "jsonrpc": "2.0",
  "id": "tx-csr",
  "result": {
    "operation": "GENERATE",
    "props": {
      "templateType": "JSONTemplate",
      "template": {
        "FriendlyName": "Запрос пользователя",
        "Description": "Сертификат для электронной подписи",
        "RDN": [
          {
            "Oid": "2.5.4.3",
            "LocalizedName": "Имя владельца",
            "DefaultValue": "Example User",
            "Length": 64,
            "ProhibitEmpty": true,
            "ProhibitChange": false
          }
        ],
        "Extensions": {
          "KeyUsage": [{ "Name": "digitalSignature", "DefaultValue": true }],
          "ExtendedKeyUsage": [{ "Name": "1.3.6.1.5.5.7.3.2", "DefaultValue": true }]
        },
        "MarkExportable": false
      }
    }
  }
}
```

Поля шаблона:

Поле	Описание
`FriendlyName`	Название шаблона в интерфейсе
`Description`	Необязательное описание
`RDN[]`	Поля владельца
`Extensions.KeyUsage[]`	Назначения ключа
`Extensions.ExtendedKeyUsage[]`	Расширенные назначения ключа, OID
`MarkExportable`	Начальное значение экспортируемости контейнера

Поля владельца

Поле `Oid` принимает как OID в точечной записи, так и распространённые сокращения: `CN`, `SN`, `GN` или `G`, `O`, `OU`, `T` или `TITLE`, `L`, `S` или `ST`, `STREET`, `C`, `E` или `EMAILADDRESS`, `OGRN`, `OGRNIP`, `SNILS`, `INN`, `INNLE`.

Для каждого поля поддерживаются:

Свойство	Описание
`DefaultValue`	Начальное значение
`LocalizedName`	Название поля в интерфейсе
`ProhibitEmpty`	Поле обязательно для заполнения
`ProhibitChange`	Значение нельзя изменить
`SettingsValues` и `ProhibitAnyValue`	Ограниченный список допустимых значений
`Length`	Максимальная длина значения

Назначения ключа

В `KeyUsage` поддерживаются: `digitalSignature`, `nonRepudiation` (синоним `contentCommitment`), `keyEncipherment`, `dataEncipherment`, `keyAgreement`, `keyCertSign`, `cRLSign`, `encipherOnly`, `decipherOnly`.

Элемент может быть строкой либо объектом с полями `Name`, `DefaultValue`, `ProhibitChange` и `LocalizedName`. В `ExtendedKeyUsage` поле `Name` содержит OID в точечной записи.

Шаблон по сертификату

Чтобы построить шаблон по существующему сертификату, передайте его в формате DER, закодированном в Base64:

```
{
  "jsonrpc": "2.0",
  "id": "tx-csr",
  "result": {
    "operation": "GENERATE",
    "props": {
      "templateType": "CertificateTemplate",
      "template": {
        "certificateBase64": "<DER Base64>"
      }
    }
  }
}
```

Поля владельца, назначения ключа и расширенные назначения исходного сертификата становятся заблокированными значениями шаблона. Экспортируемость контейнера по умолчанию выключена.

Результат

После подтверждения параметров и создания ключа приложение отправляет:

```
{
  "jsonrpc": "2.0",
  "method": "certrequests.base64",
  "params": {
    "id": "tx-csr",
    "certificaterequestBase64": "<PKCS#10 DER Base64>",
    "friendlyName": "Example User"
  }
}
```

Поле `friendlyName` берётся из общего имени (CN) итогового владельца. Для сетевого вызова результат уходит на адрес исходного вызова; локальный вызов сохраняет конверт в файл `certrequests-<id>.json` в каталоге `localResultParams.savePath`.

Смотрите также

- Описание и возможности API — общий формат вызова.
- Команда certificates — установка выпущенного сертификата.
- *Создание запроса и самоподписанного сертификата* (Руководство пользователя, глава 23) — та же операция в интерфейсе.

7 Команда diagnostics

Команда diagnostics внешнего API КриптоАРМ: версии, доступность криптопровайдеров, состояние лицензий и сведения о личных сертификатах. Формат запроса, ответа и разрешение пользователя.

Команда возвращает сведения о рабочем месте: систему, доступность криптопровайдеров, версии, состояние лицензий и сведения о личных сертификатах.

Общая информация

Возвращаются только запрошенные сведения: приложение не собирает и не отправляет сведения, которые сервис не запрашивал. Повторяющиеся значения `operation` отбрасываются, неизвестное значение отклоняется.

В запросе параметров самой команды `diagnostics` поле `diagnostic` — пустой объект: сведения о рабочем месте не передаются до того, как получены параметры и разрешение пользователя.

Формат ссылки

```
cryptoarm://diagnostics/<URL>?id=<id>
```

Формат локального вызова описан в разделе Формат ссылки для локального вызова.

Параметры

Поле `operation` — непустой массив значений:

Значение	Что возвращается
`SYSTEMINFORMATION`	Тип системы, архитектура, платформа и тип установочного пакета
`CSP_ENABLED`	Доступность КриптоПро CSP
`CADES_ENABLED`	Доступность CAdES
`VERSIONS`	Версии КриптоПро CSP, КриптоАРМ и системного OpenSSL
`PROVIDERS`	Доступность ГОСТ 2012-256, ГОСТ 2012-512 и OpenSSL
`LICENSES`	Состояние лицензий КриптоПро CSP и КриптоАРМ
`PERSONALCERTIFICATES`	Сведения о сертификатах из хранилища личных сертификатов

Поля `props.headerText` и `props.descriptionText` задают текст в окне подтверждения и ограничены 40 и 120 символами. Для локального вызова `localResultParams.savePath` принимается рядом с `operation` либо внутри `props`.

```
{
  "jsonrpc": "2.0",
  "id": "tx-diagnostics",
  "result": {
    "operation": ["SYSTEMINFORMATION", "VERSIONS", "PROVIDERS", "LICENSES"],
    "props": {
      "headerText": "Проверка рабочего места"
    }
  }
}
```

Подтверждение пользователем

Доверие к сервису само по себе не разрешает диагностику. Приложение показывает адрес сервиса и запрошенный набор сведений в окне **«Диагностика рабочего места»** и ждёт решения пользователя.

Для сетевого сервиса в этом окне можно разрешить последующие запуски диагностики в фоне. Разрешение сохраняется в профиле сервиса только вместе с подтверждением текущей операции; отказ профиль не изменяет. Изменить его позже можно на вкладке «Параметры ресурса». Пока разрешение действует, последующие запросы этого сервиса выполняются без окна подтверждения — включая `PERSONALCERTIFICATES`, который возвращает сведения обо всех личных сертификатах пользователя.

Результат

Результат отправляется методом `diagnostics.information` и содержит только запрошенные поля:

```
{
  "jsonrpc": "2.0",
  "method": "diagnostics.information",
  "params": {
    "id": "tx-diagnostics",
    "SYSTEMINFORMATION": {
      "type": "Darwin",
      "arch": "arm64",
      "platform": "darwin",
      "packageType": "pkg"
    },
    "VERSIONS": {
      "csp": "5.0.13000",
      "cryptoarm": "1.2.0",
      "openssl": "3.0.13"
    },
    "PROVIDERS": {
      "GOST2012_256": true,
      "GOST2012_512": true,
      "openssl": true
    },
    "LICENSES": {
      "csp": { "status": true, "type": "client", "expiration": "" },
      "cryptoarm": { "status": true, "type": "Permanent", "expiration": "" }
    }
  }
}
```

Поле `packageType` принимает значения `msi`, `pkg` или `deb` — по платформе, на которой запущено приложение.

Сведения о лицензии:

Поле	Описание
`status`	Лицензия действует
`type`	Для КриптоАРМ: `Trial`, `Permanent` или пустая строка. Для CSP: `client`, `server`, `standard`, `extended` или пустая строка
`expiration`	Срок действия срочной лицензии или пустая строка, если срок не задан

Формат `expiration` зависит от продукта: для КриптоАРМ это дата в формате ISO 8601 (например, `2027-01-31T00:00:00.000Z`), для КриптоПро CSP — Unix-время в миллисекундах.

Тип лицензии CSP возвращается КриптоПро на языке установленной версии CAdESCOM. Если метку не удалось сопоставить или лицензия отсутствует, приложение передаёт пустую строку.

Сведения о личных сертификатах

Значение `PERSONALCERTIFICATES` возвращает массив объектов. Закрытые ключи и содержимое сертификатов в него не входят.

Поле	Тип	Описание
`hash`	string	Отпечаток сертификата SHA-1
`serial`	string	Серийный номер
`issuerName`	string	Издатель
`issuerFriendlyName`	string	Издатель в удобочитаемом виде
`subjectName`	string	Владелец
`subjectFriendlyName`	string	Владелец в удобочитаемом виде
`organizationName`	string	Организация
`notBefore`	number	Начало срока действия, Unix-время в миллисекундах
`notAfter`	number	Окончание срока действия, Unix-время в миллисекундах
`keyNotAfter`	number	Окончание срока действия закрытого ключа
`key`	boolean	К сертификату есть закрытый ключ
`isSelfSigned`	boolean	Самоподписанный сертификат
`pubKeyAlg`	string	Алгоритм открытого ключа
`signatureAlgorithm`	string	Алгоритм подписи
`signatureDigestAlgorithm`	string	Алгоритм хеширования подписи
`keyUsageString`	string	Назначения ключа, перечисленные через запятую
`provider`	string	Криптопровайдер

Смотрите также

- Описание и возможности API — общий формат вызова.
- Приложение и пользователь — окно разрешения на диагностику.
- *Криптопровайдеры и стандарты* (Руководство пользователя, глава 2) — назначение криптопровайдеров и стандартов.

API

8 Команда startView

Команда startView внешнего API КриптоАРМ открывает нужный раздел приложения: подпись и шифрование, проверку, сертификаты, контейнеры или настройки.

Команда открывает существующий раздел приложения после обычной проверки доверия к вызову.

Формат ссылки

```
cryptoarm://startView/<URL>?id=<id>
```

Формат локального вызова описан в разделе Формат ссылки для локального вызова.

Параметры

Параметры содержат единственное обязательное поле `uiView`:

```
{
  "jsonrpc": "2.0",
  "id": "tx-view",
  "result": {
    "uiView": "SIGN_AND_ENCRYPT"
  }
}
```

Поддерживаемые значения:

uiView	Раздел приложения
`SIGN_AND_ENCRYPT`	Подпись и шифрование
`VERIFY_AND_DECRYPT`	Проверка и расшифрование
`CERTIFICATES_MY`	Личные сертификаты
`CERTIFICATES_ADDRESS_BOOK`	Сертификаты других пользователей
`CERTIFICATES_CA`, `CERTIFICATES_ROOT`	Сертификаты удостоверяющих центров; оба значения открывают один раздел
`KEYS`	Ключевые контейнеры
`ABOUT`	Настройки приложения

Пустое и неизвестное значение возвращает ошибку.

Значения `CERTIFICATES_CA` и `CERTIFICATES_ROOT` принимаются оба и открывают один и тот же раздел: отдельной вкладки для корневых сертификатов в приложении нет. Разделять промежуточные и корневые хранилища можно в команде `certificates` — там `CA` и `ROOT` задают разные хранилища.

Завершение

После открытия раздела отдельный результат не отправляется. Ошибка открытия, отказ в доверии или неподдерживаемое значение завершаются по общим правилам — см. Ошибки и отмена.

Смотрите также

- Описание и возможности API — общий формат вызова.
- *Интерфейс приложения* (Руководство пользователя, глава 9) — из каких разделов состоит приложение.

9 Лимиты, статусы и ошибки

Ограничения внешнего API КриптоАРМ: размеры файлов и JSON, таймауты, очередь вызовов, статусы Completed, Error и Canceled, поведение при отмене и недоступном сервисе.

Эта статья описывает ограничения внешнего API, статусы завершения операции и поведение приложения при ошибке и отмене.

Лимиты

Что ограничивается	Значение
Ожидающие вызовы API	8; выполняются последовательно
Один запрос JSON: параметры или результат	60 секунд
Загрузка файла и отправка результата в multipart	600 секунд
Ожидание действия пользователя	5 минут
Ответ сервиса, встроенный JSON или локальный файл	2 МиБ
Один входной файл	512 МиБ
Все входные файлы одной операции	512 МиБ и не более 100 файлов
Транспортный ZIP-архив	Не более 100 записей и 512 МиБ распакованных данных
Данные, встроенные в один JSON-результат в Base64	64 МиБ суммарно

Повторный вызов отклоняется, пока предыдущий не завершён. Вызовы различаются по источнику, команде и идентификатору транзакции, а вызов со ссылкой на файл параметров — по пути к этому файлу.

Base64 увеличивает объём передачи примерно на треть. Для больших результатов используйте `multipart/form-data` или `returnFiles: false`.

Время ожидания запроса и время ожидания действия пользователя отсчитываются независимо. Общего времени ожидания у сессии нет: прервать запущенную криптографическую операцию безопасно невозможно.

Статусы операции

`status`	Значение
`Completed`	Операция завершена; результат может описывать недействительную подпись
`Error`	Техническая или операционная ошибка

status	Значение
`Canceled`	Пользователь отказался от выполнения или отменил операцию

Для `Error` дополнительно передаются `Error: true` и `ErrorDescription`. Частичный успех сохраняет доступные результаты и сообщает ошибку для неуспешной части.

`ErrorDescription` содержит стабильные коды приложения — например, `x509.sign.failed` или `x509.verify.invalidSignature`. Сообщения криптографических библиотек в контракт не входят: они остаются в лог-файле и в журнале приложения.

> 💡 **Примечание**: недействительная подпись — это результат проверки, а не ошибка. Конверт имеет `status: "Completed"`, а `signValid` и `signers[].isValid` — значение `false`.

Ошибки и отмена

Если поддерживаемая сетевая команда уже прошла проверку доверия, приложение отправляет сообщение об ошибке, чтобы сервис не оставался в ожидании.

Если обработчик команды ещё не сформировал результат, сообщение уходит на исходный адрес методом `<команда>.parameters`:

```
{
  "jsonrpc": "2.0",
  "method": "signAndEncrypt.parameters",
  "id": "tx-42",
  "params": {
    "status": "Error",
    "Error": true,
    "ErrorDescription": "Operation failed"
  }
}
```

Ошибки самой проверки доверия, неизвестная команда и недоступный адрес могут завершиться без сообщения. Вызывающая сторона должна иметь собственный таймаут и обрабатывать повторный вызов идемпотентно.

При закрытии или перезагрузке окна операция отменяется до обращения к криптопровайдеру. Уже запущенная операция не отменяется.

Завершение вкладки операции

Если задан `uploader`, включён `returnFiles` и результат успешно доставлен, вкладка операции закрывается автоматически, а временные файлы удаляются. В остальных случаях результат остаётся во вкладке, пока пользователь её не закроет.

Смотрите также

- Описание и возможности API — общий формат вызова.
- Транспорт и доверие — проверка сервиса и адресов.
- Команда signAndEncrypt — форматы результата операции.